

28. 5. 2004

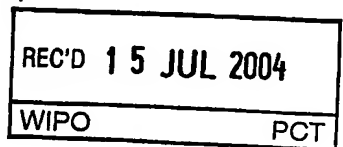
日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2 0 0 3 年 5 月 2 8 日

出 願 番 号
Application Number: 特 願 2 0 0 3 - 1 5 1 4 7 2
[ST. 10/C]: [J P 2 0 0 3 - 1 5 1 4 7 2]



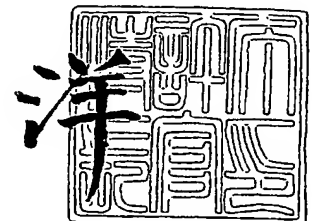
出 願 人
Applicant(s): 松下電器産業株式会社

PRIORITY
DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH RULE 17.1(a) OR (b)

2 0 0 4 年 7 月 1 日

特許庁長官
Commissioner,
Japan Patent Office

小 川



【書類名】 特許願

【整理番号】 2968250009

【提出日】 平成15年 5月28日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 9/45

【発明者】

【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

【氏名】 土井 繁則

【発明者】

【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

【氏名】 青木 博司

【発明者】

【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

【氏名】 今西 祐子

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100090446

【弁理士】

【氏名又は名称】 中島 司朗

【手数料の表示】

【予納台帳番号】 014823

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9003742

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム実行制御装置、プログラム実行制御方法

【特許請求の範囲】

【請求項 1】 バイトコード列を呼び出す呼出コードを含む 1 又は複数のバイトコード列からなるプログラムをプロセッサに実行させるプログラム実行制御装置であって、

前記プログラムの実行過程において呼出コードが実行対象とされる度に、呼び出されるべきバイトコード列が前記プロセッサ用のネイティブコードに変換済みか否かを判定する判定手段と、

前記判定手段により否定的な判定がなされた場合に、前記呼び出されるべきバイトコード列を逐次解釈して前記プロセッサに実行させ、且つ、当該バイトコード列を当該プロセッサ用のネイティブコードに変換する要求を行う第 1 手段と、

前記判定手段により肯定的な判定がなされた場合に、前記ネイティブコードをプロセッサに実行させる第 2 手段と、

前記第 1 手段によりなされるバイトコード列の逐次解釈実行又は前記第 2 手段によりなされる前記ネイティブコードの実行と並行して、前記要求に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる第 3 手段とを備える

ことを特徴とするプログラム実行制御装置。

【請求項 2】 マルチタスクオペレーティングシステムの制御下で、前記第 1 手段によりなされる前記バイトコード列の実行又は前記第 2 手段によりなされる前記ネイティブコードの実行と、前記第 3 手段によりなされる前記変換処理の実行とは、それぞれ異なるタスクとして実行され、前記第 1 手段又は前記第 2 手段のタスク実行は、前記第 3 手段のタスク実行より優先度が高いことを特徴とする請求項 1 記載のプログラム実行制御装置。

【請求項 3】 前記第 1 手段又は前記第 2 手段のタスク実行において待ち状態が発生した場合、前記第 3 手段のタスク実行に切り替える切替手段を備えることを特徴とする請求項 2 記載のプログラム実行制御装置。

【請求項 4】 前記第 1 手段によりなされる前記要求に応じて、当該要求に係

るバイトコード列をネイティブコードに変換するための情報であるコンパイル要求情報を記憶手段に登録し管理する要求管理手段を備え、

前記第3手段は、前記第1手段によりなされるバイトコード列の逐次解釈実行又は前記第2手段によりなされる前記ネイティブコードの実行と並行して、前記要求管理手段によって前記記憶手段に登録されているコンパイル要求情報に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる

ことを特徴とする請求項1～3のうちいずれか1項記載のプログラム実行制御装置。

【請求項5】 前記要求管理手段は、前記要求を先に受けたものからの順番となるキューで各コンパイル要求情報を登録して管理し、

前記第3手段は、前記キューの先頭位置にあるコンパイル要求情報から順番に前記変換処理を前記プロセッサに実行させる

ことを特徴とする請求項4記載のプログラム実行制御装置。

【請求項6】 前記要求管理手段は、前記第1手段によりなされる前記要求に係るバイトコード列のコンパイル要求情報が前記記憶手段に既に登録されている場合に、2重登録を行わないことを特徴とする請求項5記載のプログラム実行制御装置。

【請求項7】 前記要求管理手段は、

繰り返し前記要求がなされるバイトコード列については、その要求回数を計数し、計数した要求回数を当該バイトコード列のコンパイル要求情報に含めて前記記憶手段に記録する回数記録手段と、

記録した各コンパイル要求情報の要求回数を比較して、要求回数が多いもの順となるように各コンパイル要求情報のキュー位置を入替える入替手段とを備えることを特徴とする請求項5記載のプログラム実行制御装置。

【請求項8】 各バイトコード列の優先度を示す優先度情報を取得する優先度情報取得手段を更に備え、

前記要求管理手段は、

前記第1手段によりなされる前記要求に応じて、取得された優先度情報を参照

し、当該要求に係るバイトコード列の優先度を特定する特定手段と、

前記記憶手段に登録されている各コンパイル要求情報に係るバイトコード列の優先度と、前記特定手段により特定された優先度とを比較する比較手段と、

比較した結果に基づいて、優先度の高いものの順の配列となるように、前記要求に係るバイトコード列のコンパイル要求情報のキュー位置を決定する決定手段とを備える

ことを特徴とする請求項 4 記載のプログラム実行制御装置。

【請求項 9】 前記要求管理手段は、優先度の異なる複数のキューを管理しており、

前記第 3 手段は、優先度が最も高いキューに登録されているコンパイル要求情報に係るバイトコード列から順番に前記変換処理を前記プロセッサに実行させることを特徴とする請求項 5 記載のプログラム実行制御装置。

【請求項 10】 前記プログラムの実行前に、ネイティブコードに変換すべき複数のバイトコード列を示す特別要求情報を取得する特別要求情報取得手段を更に備え、

前記要求管理手段は、

取得された特別要求情報に示される各バイトコード列のコンパイル要求情報を、優先度が最も高いキューに一括登録し、管理する

ことを特徴とする請求項 9 記載のプログラム実行制御装置。

【請求項 11】 各バイトコード列の優先度を示す優先度情報を取得する優先度情報取得手段と、

前記変換処理を行うバイトコード列の優先度と前記命令実行タスクの優先度とを比較する比較手段と、

比較した結果、前記バイトコード列の優先度の方が命令実行タスクの優先度より高い場合に、前記コンパイルタスクの優先度を一時的に高くする優先度変更手段とを更に備える

ことを特徴とする請求項 2 記載のプログラム実行制御装置。

【請求項 12】 あるバイトコード列と関連する関連バイトコード列を対応付けている対応情報を取得する対応情報取得手段と、

前記対応情報を参照して、前記要求に係るバイトコード列と対応付けられている関連バイトコード列があるか否かを検索する検索手段とを更に備え、

前記検索手段による検索結果、関連バイトコード列が検出された場合、前記要求管理手段は、検出された関連バイトコード列のコンパイル要求情報を記憶手段に登録する

ことを特徴とする請求項 4 記載のプログラム実行制御装置。

【請求項 13】 前記判定手段により否定的な判定がなされたときに、前記バイトコード列が現在、前記変換処理の途中であるか否かを判定する第 2 判定手段と、

前記第 2 判定手段により肯定的な判定がなされた場合に、当該変換処理が終了するのを待って、変換されたネイティブコードを前記プロセッサに実行させる第 4 手段とを更に備える

ことを特徴とする請求項 1 記載のプログラム実行制御装置。

【請求項 14】 各バイトコード列の優先度を示す優先度情報を取得する優先度情報取得手段を更に備え、

前記第 3 手段は、前記記憶手段に複数登録されているコンパイル要求情報に係るバイトコード列のうち、取得された前記優先度情報に基づいて優先度の高いものから順番に前記変換処理を前記プロセッサに実行させる

ことを特徴とする請求項 1～3 のうちいずれか 1 項記載のプログラム実行制御装置。

【請求項 15】 前記要求管理手段は、

閾値を取得する取得手段と、

繰り返し前記要求がなされるバイトコード列については、その要求回数を計数し、当該バイトコード列のコンパイル要求情報に含めて計数した要求回数を前記記憶手段に登録する回数記録手段とを備え、

前記第 3 手段は、前記要求回数が閾値を超えたものから順番に前記変換処理を前記プロセッサに実行させる

ことを特徴とする請求項 1～3 のうちいずれか 1 項記載のプログラム実行制御装置。

【請求項 16】 バイトコード列を呼び出す呼出コードを含む 1 又は複数のバイトコード列からなるプログラムをプロセッサに実行させるプログラム実行制御装置におけるプログラム実行制御方法であって、

前記プログラムの実行過程において呼出コードが実行対象とされる度に、呼び出されるべきバイトコード列が前記プロセッサ用のネイティブコードに変換済みか否かを判定する判定ステップと、

前記判定ステップにおいて否定的な判定がなされた場合に、前記呼び出されるべきバイトコード列を逐次解釈して前記プロセッサに実行させ、且つ、当該バイトコード列を当該プロセッサ用のネイティブコードに変換する要求を行う第 1 ステップと、

前記判定ステップにおいて肯定的な判定がなされた場合に、前記ネイティブコードをプロセッサに実行させる第 2 ステップと、

前記第 1 ステップにおいてなされるバイトコード列の逐次解釈実行又は前記第 2 ステップにおいてなされる前記ネイティブコードの実行と並行して、前記要求に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる第 3 ステップとを含む

ことを特徴とするバイトコード実行制御方法。

【請求項 17】 バイトコード列を呼び出す呼出コードを含む 1 又は複数のバイトコード列からなるプログラムをプロセッサに実行させるプログラム実行制御装置にプログラム実行制御処理を行わせるための制御プログラムであって、

前記プログラム実行制御処理は、

前記プログラムの実行過程において呼出コードが実行対象とされる度に、呼び出されるべきバイトコード列が前記プロセッサ用のネイティブコードに変換済みか否かを判定する判定ステップと、

前記判定ステップにおいて否定的な判定がなされた場合に、前記呼び出されるべきバイトコード列を逐次解釈して前記プロセッサに実行させ、且つ、当該バイトコード列を当該プロセッサ用のネイティブコードに変換する要求を行う第 1 ステップと、

前記判定ステップにおいて肯定的な判定がなされた場合に、前記ネイティブコ

ードをプロセッサに実行させる第2ステップと、

前記第1ステップにおいてなされるバイトコード列の逐次解釈実行又は前記第2ステップにおいてなされる前記ネイティブコードの実行と並行して、前記要求に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる第3ステップとを含む

ことを特徴とする制御プログラム。

【請求項18】 バイトコード列を呼び出す呼出コードを含む1又は複数のバイトコード列からなるプログラムをプロセッサに実行させるプログラム実行制御装置にプログラム実行制御処理を行わせるための制御プログラムを記録した記録媒体であって、

前記プログラム実行制御処理は、

前記プログラムの実行過程において呼出コードが実行対象とされる度に、呼び出されるべきバイトコード列が前記プロセッサ用のネイティブコードに変換済みか否かを判定する判定ステップと、

前記判定ステップにおいて否定的な判定がなされた場合に、前記呼び出されるべきバイトコード列を逐次解釈して前記プロセッサに実行させ、且つ、当該バイトコード列を当該プロセッサ用のネイティブコードに変換する要求を行う第1ステップと、

前記判定ステップにおいて肯定的な判定がなされた場合に、前記ネイティブコードをプロセッサに実行させる第2ステップと、

前記第1ステップにおいてなされるバイトコード列の逐次解釈実行又は前記第2ステップにおいてなされる前記ネイティブコードの実行と並行して、前記要求に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる第3ステップとを含む

ことを特徴とする記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、バイトコード列からなるプログラムをプロセッサに実行させるプロ

グラム実行制御装置に関する。

【0002】

【従来の技術】

特定のプラットフォームに依存することなくプログラムの実行を実現するソフトウェアである仮想マシンが、組み込み機器等のプロセッサを備えた様々な家電製品に実装されている。代表的な仮想マシンとして、Java（登録商標）バイトコード（以下、単にバイトコードという。）を解釈して実行するJVM（Java Virtual Machine）が挙げられる。JVMの仕様については、下記の非特許文献に詳しく説明されている。

【0003】

上述のバイトコードは、一般的にはJava言語で記述されたソースプログラムをjavac等のJavaコンパイラで変換することにより生成され、.class拡張子のクラスファイルと呼ばれるファイルに格納される。

JVMは、クラスファイルを解釈して、当該クラスファイルに含まれている各種メソッドをプロセッサに実行させるための制御を行う。メソッドとは、バイトコード列から成る特定の処理を行う手続きのことをいい、他のプログラミング言語ではメンバ関数と呼ばれているものである。

【0004】

基本的にJVMは、バイトコードを逐次解釈してプロセッサに処理を実行させるインタプリタとして機能するものであるが、一般的に言われているようにその実行速度は、プロセッサが直接理解できるネイティブコードに変換されたプログラムの実行速度に比べると遅い。特に、同じメソッドが繰り返し呼び出されるようなプログラムの場合、呼び出される度に同じメソッドを解釈することになるので、無駄が多い。

【0005】

そのため、プログラム実行過程においてメソッドを呼び出した時に、そのメソッドが未コンパイルであるか否かを判定し、未コンパイルであれば、当該メソッドをネイティブコードに変換するJiT（Just in Time）コンパイラを備えるJVMが考案されている。

これにより、同じメソッドが再び呼び出されるような場合、先のメソッド呼び出し時に変換したネイティブコードを実行すればよいので、実行速度を速めることができる。

【0006】

また、ネイティブコードに変換すべきメソッドを特定するためのJ i Tコンパイラの拡張技術として、J a v a H o t S p o t（登録商標）と呼ばれる技術がある。これは、プログラムの実行過程において、呼び出したメソッドの呼出回数を集計し、呼出回数が閾値を超えた時に当該メソッドをネイティブコードに変換する技術である。

【0007】

J i Tコンパイラ付きJVMのように、インタプリタ機能とコンパイラ機能を共に備えるプログラム実行装置が、下記の特許文献1、特許文献2に示されている。

【0008】

【非特許文献】

J a v a 仮想マシン仕様 第2版（株式会社ピアソン・エデュケーション発行、ISBN4-89471-356-X）

【0009】

【特許文献1】

特開平4-178734号公報

【0010】

【特許文献2】

特開昭64-62705号公報

【0011】

【発明が解決しようとする課題】

ところで、上述したJ i Tコンパイラ付きJVMは、呼び出したメソッドが未コンパイルであれば直ちにコンパイルを開始し、コンパイルが終了するのを待って、生成されたネイティブコードをプロセッサに実行させるので、未コンパイルのメソッドを実行する時は、同じメソッドをインタプリタで実行するより実行速

度が遅くなるという問題があった。また、Java HotSpot 技術も同様に、未コンパイルのメソッドを呼び出した時にコンパイルを行う場合は、実行速度が遅くなる。

【0012】

特に、プログラムの実行開始当初にコンパイル処理が頻発するので実行速度の低下が顕著に現れる。

本発明は、上述した問題を解決するべく、未コンパイルのメソッド実行時に当該メソッドをコンパイルすることにより生じていた実行速度の低下を解消するプログラム実行制御装置、プログラム実行制御方法を提供することを目的とする。

【0013】

【課題を解決するための手段】

上記目的を達成するために本発明に係るプログラム実行制御装置は、バイトコード列を呼び出す呼出コードを含む1又は複数のバイトコード列からなるプログラムをプロセッサに実行させるプログラム実行制御装置であって、前記プログラムの実行過程において、呼出コードが実行対象とされる度に、呼び出されるべきバイトコード列が前記プロセッサ用のネイティブコードに変換済みか否かを判定する判定手段と、前記判定手段により否定的な判定がなされた場合に、前記呼び出されるべきバイトコード列を逐次解釈して前記プロセッサに実行させ、且つ、当該バイトコード列を当該プロセッサ用のネイティブコードに変換する要求を行う第1手段と、前記判定手段により肯定的な判定がなされた場合に、前記ネイティブコードをプロセッサに実行させる第2手段と、前記第1手段によりなされるバイトコード列の逐次解釈実行又は前記第2手段によりなされる前記ネイティブコードの実行と並行して、前記要求に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる第3手段とを備えることを特徴とする。

【0014】

また、本発明に係るプログラム実行制御方法は、バイトコード列を呼び出す呼出コードを含む1又は複数のバイトコード列からなるプログラムをプロセッサに実行させるプログラム実行制御装置におけるプログラム実行制御方法であって、

前記プログラムの実行過程において、呼出コードが実行対象とされる度に、呼び出されるべきバイトコード列が前記プロセッサ用のネイティブコードに変換済みか否かを判定する判定ステップと、前記判定ステップにおいて否定的な判定がなされた場合に、前記呼び出されるべきバイトコード列を逐次解釈して前記プロセッサに実行させ、且つ、当該バイトコード列を当該プロセッサ用のネイティブコードに変換する要求を行う第1ステップと、前記判定ステップにおいて肯定的な判定がなされた場合に、前記ネイティブコードをプロセッサに実行させる第2ステップと、前記第1ステップにおいてなされるバイトコード列の逐次解釈実行又は前記第2ステップにおいてなされる前記ネイティブコードの実行と並行して、前記要求に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる第3ステップとを含むことを特徴とする。

【0015】

ここで、バイトコード列は、例えば、メソッド、又はメンバ関数を指す。

これらの構成により、呼び出すバイトコード列が未コンパイルであれば、当該バイトコード列を逐次解釈して実行するので、従来のJITコンパイラ付JVMが未コンパイルのメソッド実行時に当該メソッドをコンパイルすることにより生じていた、プログラム実行開始当初に顕著に現れる実行速度の低下を解消することができ、また、並行して未コンパイルメソッドのコンパイルも行うので、全体的なプログラム実行速度が低下することもない。

【0016】

また、本発明は、上記プログラム実行制御方法の各ステップを含むプログラム実行制御処理をプログラム実行制御装置に行わせる制御プログラムであってもよく、当該制御プログラムを記録した記録媒体であってもよい。

【0017】

【発明の実施の形態】

本発明に係るプログラム実行制御装置は、メモリ又はハードディスク等の記憶媒体と、プロセッサを備えるコンピュータ装置であり、具体的には携帯電話機や、デジタル放送受信装置等である。

プログラム実行制御装置が有する記憶媒体にはマルチタスクOS (Opera

ting System)、JVM、クラスファイル、そして、よく使われる汎用的なクラス又はメソッド等の部品群から成るクラスライブラリ等が格納されており、本プログラム実行制御装置の各種機能は、記憶媒体に格納されているマルチタスクOS、及び当該マルチタスクOSの制御下において動作するJVMに従ってプロセッサが動作することにより実現される。

【0018】

本プログラム実行制御装置は、プログラムの実行過程においてメソッドを呼び出す場合、当該メソッドが未コンパイルであるか否かを判定し、未コンパイルであれば、当該メソッドのバイトコードをインタプリタ処理で実行すると共に当該メソッドをコンパイルする要求を登録し、コンパイル済であれば、当該メソッドのネイティブコードを実行する。そして、登録されたコンパイル要求に係るメソッドのコンパイルは、バイトコードのインタプリタ処理による実行又はネイティブコードの実行といった命令実行を行うタスクとは別のタスクで行う。

【0019】

つまり、バイトコード又はネイティブコードの実行を行うタスクを命令実行タスク、コンパイル処理を行うタスクをコンパイルタスクと呼ぶとすると、命令実行タスクとコンパイルタスクを非同期で並行処理している点が、本発明の特徴である。

図1は、命令実行タスクとコンパイルタスクの実行状態推移を示す図である。

【0020】

本プログラム実行制御装置は、プリエンプティブで優先度ベースのタスクスケジューリングによりシングルプロセッサ資源の割り当てを行っている。

各タスクの優先度は、命令実行タスクを高とし、コンパイルタスクを低としており、優先度高のタスクに割り当てられるプロセッサ時間配分と優先度低のタスクに割り当てられるプロセッサ時間配分の比率は、9：1としている。また、命令実行タスクの実行において待ち状態が発生した場合、直ちにコンパイルタスクの実行に切り替える制御を行う。

【0021】

これにより、本発明のプログラム実行制御装置は、未コンパイルのメソッドを

呼び出したタイミングではコンパイル処理を実行せず、コンパイルタスクの実行時にコンパイルするので、プログラムの実行開始当初に頻発するコンパイル処理に伴う立ち上がりの遅延を抑えることができ、実行速度の低下を抑えることができる。

【0022】

以下、本発明に係るプログラム実行制御装置の各実施形態について説明する。

<実施形態1>

<構成1>

図2は、実施形態1のプログラム実行制御装置1の機能構成図である。

プログラム実行装置1は、同図に示すように、クラスファイル/クラスライブラリ記憶部2、命令実行処理部3、コンパイル要求管理部4、ネイティブコード記憶部5、コンパイル処理部6、コンパイル済メソッド情報テーブル7及びマルチタスク制御部19を備える。なお、同図には本発明の特徴となる機能部のみを示しており、図示していない、マルチタスクOS及びJVMが実装されているコンピュータ装置が通常有する各機能については説明を省略する。

【0023】

クラスファイル/クラスライブラリ記憶部2は記憶媒体であり、クラスファイル及びクラスライブラリが格納されている。

クラスファイルは、アンテナ21又はネットワーク20より取得することができる。

命令実行処理部3は、クラスファイル/クラスライブラリ記憶部2から読み出したバイトコードを逐次解釈してプロセッサに実行させ、メソッド呼び出し時に当該メソッドがコンパイル済みであれば、当該メソッドのネイティブコードをプロセッサに実行させる処理を行う。

【0024】

ここで、命令実行処理部3に含まれる各機能について説明する。命令実行処理部3には、クラスロード部8、命令取出部9、命令解読部10、命令実行部11、メソッド種別判定部12、及びネイティブコード取出部13が含まれる。

クラスロード部8は、クラスファイル/クラスライブラリ記憶部2に格納され

ているクラスファイル又はクラスライブラリをメモリにロードする機能を有する。

【0025】

命令取出部 9 は、クラスロード部 8 によってメモリにロードされたクラスファイル又はクラスライブラリを構成するバイトコードを命令単位で取り出す機能を有する。

命令解読部 10 は、命令取出部 9 によって取り出された命令単位のバイトコードを解読してネイティブコードを選択する機能を有する。

【0026】

命令実行部 11 は、ネイティブコードをプロセッサに実行させる機能を有する。

命令実行部 11 により実行される命令が、メソッドを呼び出す呼出コードである場合、メソッド種別判定部 12 は、コンパイル済メソッド情報テーブル 7 を参照して、当該呼出コードによって呼び出されるべきメソッドがコンパイル済みか否かを判定する機能を有する。呼出コードは、J a s m i n のアセンブラコード表記で言えば、`invokevirtual`、`invokespecial`、`invokestatic`、`invokeinterface` といったインストラクションに相当する。

【0027】

呼び出されるべきメソッドがコンパイル済みと判定された場合、ネイティブコード取出部 13 は、該当のメソッドのネイティブコードをネイティブコード記憶部 5 から取り出して命令実行部 11 に送る。命令実行部 11 は、取り出されたネイティブコードをプロセッサに実行させる。

呼び出されるべきメソッドが未コンパイルと判定された場合、メソッド種別判定部 12 は、当該メソッドをコンパイルする要求をコンパイル要求管理部 4 に通知する。そして、命令取出部 9 は該当メソッドのバイトコード列を命令単位で逐次取り出し、命令解読部 10 は取り出されたバイトコード列を命令単位で逐次解読し、命令実行部 11 は逐次解釈された結果の命令単位のネイティブコードをプロセッサに実行させる。

【0028】

コンパイル要求管理部 4 は、メソッド種別判定部 12 からコンパイル要求の通知を受けて、受け付けたコンパイル要求に係るメソッドをコンパイルする順番について管理する機能を有する。コンパイル要求管理部 4 は、登録要求受付部 14、キュー制御部 15 及びキュー記憶部 16 を有する。

登録要求受付部 14 は、メソッド種別判定部 12 からコンパイル要求の通知を受け付ける機能を有する。この時に、メソッドに関する情報、例えば、メソッドを特定する識別情報や、メソッドを構成するバイトコード列の格納先アドレス等を取得する。

【0029】

キュー記憶部 16 は RAM 等の記憶媒体である。

キュー制御部 15 は、キュー記憶部 16 に複数のコンパイル要求情報を記憶させ、これらのコンパイル要求情報をキューで管理する。詳しくは、登録要求受付部 14 において受け付けたコンパイル要求をコンパイル要求情報としてキュー記憶部 16 に登録し、コンパイル処理部 6 からコンパイル要求情報の取得要求を受け付けると、キューの先頭位置にあるコンパイル要求情報が示すメソッドをコンパイル処理部 6 に通知し、コンパイル処理部 6 においてメソッドのコンパイルが終了するとコンパイル処理部 6 から通知を受けて、当該メソッドのコンパイル要求情報をキュー記憶部 16 から削除する。

【0030】

キュー制御部 15 は、コンパイル処理部 6 からコンパイル要求情報の取得要求を受けて、キューの先頭位置にあるコンパイル要求情報が示すメソッドをコンパイル処理部 6 に通知する際、コンパイル要求情報に含まれるコンパイル状態を示すビットフラグを立てて、コンパイル中であることを記録する。

キュー制御部 15 は、コンパイル要求情報の登録と、登録されているコンパイル要求情報の削除とを排他制御で行う。即ち、同時にコンパイル要求情報の登録と削除を行わないようにしている。また、登録しようとしているコンパイル要求情報がキュー記憶部 16 に既に登録されていれば、登録は行わない。

【0031】

コンパイル処理部 6 は、コンパイル要求取得部 17 及びメソッドコンパイル部 18 から構成されており、コンパイル要求取得部 17 は、キュー制御部 15 にコンパイルすべきメソッドの情報を要求して取得する機能を有し、メソッドコンパイル部 18 は、当該メソッドのバイトコード列をネイティブコードに変換する機能を有する。

【0032】

メソッドコンパイル部 18 において生成されたネイティブコードは、ネイティブコード記憶部 5 に格納され、メソッド名とネイティブコードの格納先アドレスとを示すコンパイル済メソッド情報が、コンパイル済メソッド情報テーブル 7 に追加される。また、コンパイル要求取得部 17 は、メソッドのコンパイルが終了すると、当該メソッドのコンパイルが終了した旨をキュー制御部 15 に通知する。

【0033】

マルチタスク制御部 19 は、マルチタスク OS の機能であり、優先度ベースのタスクスケジューリングに基づいて、命令実行処理部 3 により実行される命令実行タスクとコンパイル処理部 6 により実行されるコンパイルタスクそれぞれにプロセッサ資源の割り当てを行う。

<データ 1>

ここで各種データについて説明する。

【0034】

まず、プログラム実行制御装置 1 により実行されるプログラムについて説明する。

図 3 は、Java 言語で記述されたソースプログラムの一例を示す図である。なお、本発明の説明上必要がないため m1 メソッド、m2 メソッドの内容については記載を省略している。同図に示すソースプログラムを変換して生成されたバイトコードから成るクラスファイル A が、プログラム実行制御装置 1 のクラスファイル/クラスライブラリ記憶部 2 に記憶される。

【0035】

クラスファイルをプログラム実行制御装置 1 が実行する場合、その実行過程に

においてクラスファイル内の呼出コードによってメソッドが呼び出される。例えば、図3のに示すプログラムの場合だと、まず `init` メソッドが1回呼び出され、続いて、`m1` メソッドが10回呼び出され、その後、`m2` メソッドが5回呼び出される。

【0036】

次に、コンパイル要求情報について説明する。

図4は、キュー記憶部16に登録されているコンパイル要求情報の具体例を示す図である。

コンパイル要求情報は、次のコンパイル要求情報の位置を示すポインタと、メソッド情報と、コンパイル状態を示すビットフラグから成り、各コンパイル要求情報はポインタによって繋がっており、キューを構成している。

【0037】

メソッド情報には、コンパイルするメソッドを識別する情報や、メソッドを構成するバイトコード列の格納アドレス等が含まれる。

コンパイル状態を示すビットフラグは、フラグが立っていればコンパイル中、立っていなければ未コンパイルを表す。同図には、わかりやすいように「コンパイル中」、「未コンパイル」と表現している。

【0038】

同図では、`init` メソッドのコンパイル要求情報がキューの先頭となっており、続いて、`m1` メソッドのコンパイル要求情報、`m2` メソッドのコンパイル要求情報の順番で配列されている。よって、この場合、コンパイル処理部6からコンパイルすべきメソッドの情報を要求する通知がキュー制御部15に対してなされると、`init` メソッドの情報がコンパイル処理部6に対して通知される。コンパイル処理部6において、`init` メソッドのコンパイルが行われ、コンパイルが終了すると、`init` メソッドのコンパイルが終了した旨の通知をキュー制御部15に対して行う。

【0039】

キュー制御部15は、`init` メソッドのコンパイルが終了した旨の通知を受けると、`init` メソッドのコンパイル要求情報をキュー記憶部16から削除し

、キューの先頭のコンパイル要求情報を示すポインタは、m1 メソッドのコンパイル要求情報を示すように更新される。

次にコンパイル済メソッド情報について説明する。

【0040】

図5は、コンパイル済メソッド情報テーブル7の一例を示す図である。コンパイル済メソッド情報は、コンパイル済みのメソッドを特定する情報（例えば、メソッド名）と、ネイティブコードの格納先アドレスから成る。

<動作1>

ここでプログラム実行制御装置1の動作について説明する。

【0041】

図6は、命令実行処理部3により実行される命令実行タスクのフローを示す図である。

まず、命令実行部11は、取り出された命令（ネイティブコード）をプロセッサに実行させる（ステップS1）。実行する命令がプログラム終了命令であれば（ステップS2：YES）、動作を終了する。実行する命令がプログラム終了命令でなければ（ステップS2：NO）ステップS3に進む。

【0042】

ステップS3において、実行する命令が呼出コードであれば（ステップS3：YES）、ステップS4に進む。ステップS3において、実行する命令が呼出コードでなければ（ステップS3：NO）、ステップS1に戻る。

ステップS4では、メソッド種別判定部12が、コンパイル済メソッド情報テーブルを参照して呼出対象のメソッドがコンパイル済みか否かを判定する。呼出対象のメソッドがコンパイル済であれば（ステップS4：YES）、ステップS5に進む。呼出対象のメソッドがコンパイル済でなければ（ステップS4：NO）、ステップS6に進む。

【0043】

ステップS5では、ネイティブコード取出部13が該当メソッドのネイティブコードをネイティブコード記憶部5から取り出し、取り出したネイティブコードを命令実行部11に送る。その後、ステップS1に戻る。

ステップS6では、メソッド種別判定部12は、コンパイル要求をコンパイル要求管理部4に通知する。続いて、ステップS7では、命令取出部9が、該当メソッドのバイトコードを取り出し、命令解読部10が、取り出したバイトコードの解読を行う。その後、ステップS1に戻る。

【0044】

一方、コンパイル処理部6により実行されるコンパイルタスクは、マルチタスク制御部19によりプロセッサ資源が割り当てられると、キュー制御部15に要求し取得したメソッドのバイトコード列のコンパイルを行う。

続いて動作の具体的な例を図2、図4、図5を用いて説明する。図2に示す命令実行部11が、メソッドm2を呼び出す呼出コードをプロセッサに実行させる場合、メソッド種別判定部12は、当該メソッドm2がコンパイル済か否かを図5に示すコンパイル済メソッド情報テーブル7を参照して判定する。

【0045】

コンパイル済メソッド情報テーブル7には、メソッドm2を特定する情報がコンパイル済みメソッドとしてネイティブコードの格納先アドレスと共に記載されているので、メソッド種別判定部12は、当該メソッドm2がコンパイル済であると判定し、ネイティブコード取出部13に該当のネイティブコードを取り出させる。取り出されたネイティブコードは命令実行部11に送られて、実行される。

【0046】

一方、図示されていないメソッドm3を呼び出す呼出コードを命令実行部11がプロセッサに実行させる場合、メソッド種別判定部12は、図5に示すコンパイル済メソッド情報テーブル7にメソッドm3が記載されていないので未コンパイルと判定し、当該メソッドのコンパイル要求をコンパイル要求管理部4に通知する。そして、当該メソッドのバイトコードを命令単位で逐次取り出して解釈実行する、いわゆるインタプリタ処理を行う。

【0047】

コンパイル要求管理部4は、図4に示すコンパイル要求情報のキューの最後列、すなわち、メソッドm2のコンパイル要求情報の後に、メソッドm3のコンパ

イル要求情報を登録する。

＜実施形態 2＞

実施形態 2 のプログラム実行制御装置は、登録されているコンパイル要求情報のメソッドと同じメソッドについてコンパイル要求がなされる場合、コンパイル要求を受けた回数を要求回数としてコンパイル要求情報毎にカウントして記録し、登録された順番で配列されている各コンパイル要求情報の配列順を、要求回数が多いもの順に入替える順番入替処理を行う。

【0048】

これにより、頻繁に呼び出されるメソッドを先にコンパイルさせることができ、プログラム実行速度の向上に繋がる。

以下、実施形態 2 のプログラム実行制御装置について、実施形態 1 で説明したプログラム実行制御装置 1 とは異なる部分のみを、構成、データ、動作の各項に分けて説明する。

【0049】

＜構成 2＞

図 7 は、実施形態 2 のプログラム実行制御装置 1 A の機能構成図である。

実施形態 1 のプログラム実行制御装置 1 と異なる点は、キュー制御部 15 A 内に加算部 22 を備えている点である。

加算部 22 は、既に登録されているコンパイル要求情報のメソッドと同一のメソッドのコンパイル要求を受けた場合に、当該コンパイル要求情報のメソッドが未コンパイル状態であれば、当該コンパイル要求情報に含めて記録されている要求回数に 1 加算した値を算出する機能を有する。キュー制御部 15 A は、算出した値を要求回数として更新記録する。

【0050】

＜データ 2＞

次に、実施形態 2 のキュー記憶部 16 A に登録されている具体的なコンパイル要求情報について説明する。

図 8 は、キュー記憶部 16 A に登録されているコンパイル要求情報の具体例を示す図である。実施形態 1 で説明したコンパイル要求情報と異なる点は、各コン

パイル要求情報が2つのポインタを有し、配列位置の前後関係を2つのポインタで示すようにしている点と、要求回数を記録する領域を設けている点である。

【0051】

同図は、順番入替処理によって要求回数が多いもの順に入替えが行われた後の配列状態を示している。

＜動作2＞

次に、キューの順番入替処理の動作について説明する。

図9は、キュー制御部15Aが行うコンパイル要求情報の順番入替処理のフローを示す図である。

【0052】

まず、登録要求受付部14でコンパイル要求が受け付けられると、キュー制御部15Aは、キューの先頭のコンパイル要求情報を参照する（ステップS11）。

続いて、受け付けられたコンパイル要求に係るメソッドと、参照したコンパイル要求情報に係るメソッドとが同一であれば（ステップS12：YES）、ステップS13に進む。同一でなければ（ステップS12：NO）、ステップS17に進む。

【0053】

ステップS17において、全てのコンパイル要求情報を参照し終わっていれば（ステップS17：YES）、ステップS19に進み、まだ、全てのコンパイル要求情報を参照していなければ（ステップS17：NO）、ステップS18に進む。

ステップS18において、参照したコンパイル要求情報のポインタが示す、1つ後の配列位置のコンパイル要求情報を参照し、ステップS12に進む。

【0054】

ステップS19において、受け付けたコンパイル要求に係るメソッドのコンパイル要求情報を、キューの最後列に登録する。

ステップS13において、参照したコンパイル要求情報のメソッドが未コンパイルであるかコンパイル中かをビットフラグで確認する。コンパイル中であれば

(ステップ S13: NO)、入替処理を終了し、未コンパイルであれば (ステップ S13: YES)、ステップ S14 に進む。

【0055】

ステップ S14 において、参照したコンパイル要求情報の要求回数に 1 加算した値を要求回数として更新する。続いて、ステップ S15 において、1 つ前の配列位置にあるコンパイル要求情報の要求回数と、更新した要求回数の大きさを比較する。

その結果、1 つ前の配列位置にあるコンパイル要求情報の要求回数より、更新した要求回数の方が大きな値であれば (ステップ S15: YES)、ステップ S16 に進む。そうでなければ (ステップ S15: NO)、順番入替処理を終了する。

【0056】

ステップ S16 において、1 つ前の配列位置にあるコンパイル要求情報と、要求回数を更新したコンパイル要求情報の配列順番を入替える。すなわち、入替える 2 つのコンパイル要求情報及びその前後に位置する各コンパイル要求情報のポインタ情報の変更を行う。その後、順番入替処理を終了する。

<実施形態 3>

実施形態 3 のプログラム実行制御装置は、各メソッドの優先度が記載されている優先度情報テーブルをクラスファイルから取得し、取得した優先度情報テーブルを参照して、コンパイルするメソッドの優先度を特定し、優先度に基づいてコンパイル要求情報の配列位置を決定する処理を行う。

【0057】

これにより優先度の高いメソッドからコンパイルされるので、プログラム実行速度の向上に繋がる。

以下、実施形態 3 のプログラム実行制御装置について、実施形態 1 で説明したプログラム実行制御装置 1 とは異なる部分のみを、構成、データ、動作の各項に分けて説明する。

【0058】

<構成 3>

図10は、実施形態3のプログラム実行制御装置1Bの機能構成図である。

実施形態1のプログラム実行制御装置1と異なる点は、優先度情報テーブル23を取得する点と、キュー制御部15Bが、メソッドの優先度に基づいて登録するコンパイル要求情報の、キューにおける登録位置を決定している点である。

【0059】

<データ3>

次に各種データについて説明する。

図11は、クラスファイルから取得した優先度情報テーブル23の具体的な内容を示す図である。優先度情報テーブル23では、メソッドと優先度とが対応付けられている。ここでいう優先度とは、コンパイル処理を優先的に行うべき度合いを示す指標であり、値が大きい方が優先度の高いものとしている。よって、優先度の高いものから並べると、メソッドm3、メソッドinit、メソッドm2、メソッドm1の順番になる。なお、本実施形態では、各メソッドの優先度はそれぞれ異なる値となっている。

【0060】

図12は、キュー記憶部16Bに登録されているコンパイル要求情報の具体例を示す図である。同図に示すように、優先度が高いコンパイル要求情報が先になるように配列されている。

<動作3>

次にキュー制御部15Bによるコンパイル要求情報の登録位置決定処理について説明する。

【0061】

図13は、キュー制御部15Bが行うコンパイル要求情報の登録位置決定処理のフローを示す図である。

まず、登録要求受付部14Bにおいてコンパイル要求が受け付けられると、キュー制御部15Bは、キューの先頭のコンパイル要求情報を参照する（ステップS21）。

【0062】

続いて、受け付けられたコンパイル要求に係るメソッドと、参照したコンパイ

ル要求情報に係るメソッドとが同一であれば（ステップS22：YES）、登録位置決定処理を終了する。同一でなければ（ステップS22：NO）、ステップS23に進む。

ステップS23において、登録するメソッドの優先度と、参照したコンパイル要求情報に記載されている優先度とを比較し、登録するメソッドの優先度の方が高ければ（ステップS23：YES）、ステップS24に進む。参照したコンパイル要求情報に記載されている優先度の方が高ければ、（ステップS23：NO）、ステップS25に進む。

【0063】

ステップS25において、全てのコンパイル要求情報を参照し終わっていれば（ステップS25：YES）、ステップS26に進み、まだ、全てのコンパイル要求情報を参照していなければ（ステップS25：NO）、ステップS27に進む。

ステップS27において、参照したコンパイル要求情報のポインタが示す、1つ後の配列位置のコンパイル要求情報を参照し、ステップS22に進む。

【0064】

ステップS26において、受け付けたコンパイル要求に係るメソッドのコンパイル要求情報を、キューの最後列に登録して、登録位置決定処理を終了する。

ステップS24において、登録するメソッドのコンパイル要求情報を、参照したコンパイル要求情報の1つ前の配列位置に登録して、登録位置決定処理を終了する。

【0065】

例えば、図12に示すコンパイル要求情報のキューに、優先度2のメソッドm3のコンパイル要求情報を登録する場合、登録位置決定処理により、メソッドm2のコンパイル要求情報とメソッドm1のコンパイル要求情報との間にメソッドm3のコンパイル要求情報が登録されることになるが、具体的には、メソッドm2のコンパイル要求情報及びメソッドm1のコンパイル要求情報のポインタそれぞれが、メソッドm3のコンパイル要求情報の登録位置を示すように更新することで、メソッドm3のコンパイル要求情報の登録位置が決定される。

＜実施形態 4＞

実施形態 4 のプログラム実行制御装置は、優先度が高、中、低と付けられている 3 つのキューを設けて、優先度の高いキューに登録されているコンパイル要求情報のメソッドから順番にコンパイルする。すなわち、優先度高のキューの先頭位置に登録されているコンパイル要求情報のメソッドから順番にコンパイルをし、優先度高のキューに登録されているコンパイル要求情報のメソッドが全てコンパイルされると、次に優先度中のキューの先頭に登録されているコンパイル要求情報のコンパイルを行い、優先度中のキューに登録されているコンパイル要求情報のメソッドが全てコンパイルされると、次に優先度低のキューの先頭に登録されているコンパイル要求情報のコンパイルを行う。

【0066】

優先度高のキューには、プログラム実行制御装置の OS 起動と共にキュー記憶部に一括登録されるコンパイル要求情報が登録され、優先度中のキューには、優先度情報テーブルに優先度が示されているメソッドのコンパイル要求情報が登録され、優先度低のキューには、優先度高のキュー及び優先度中のキューに登録されるメソッド以外のメソッドが登録される。

【0067】

以下、実施形態 4 のプログラム実行制御装置について、実施形態 1 から実施形態 3 までで説明したプログラム実行制御装置とは異なる部分のみを、構成、データの各項に分けて説明する。

＜構成 4＞

図 14 は、実施形態 4 のプログラム実行制御装置 1 C の機能構成図である。

【0068】

実施形態 1 のプログラム実行制御装置 1 と異なる点は、キュー制御部 15 C が優先度の異なる 3 つのキューを制御している点と、一括登録部 25 を新たに備えて、優先的にコンパイルすべきメソッドのメソッド名や格納先等を示すメソッド情報のテーブルである一括登録情報 24 に基づいて、OS 起動と共にキュー記憶部 16 C の優先度高のキューに、これらのメソッドのコンパイル要求情報を登録する点と、実施形態 2 で説明した順番入替処理や、実施形態 3 で説明した登録位

置決定処理を行ってコンパイル要求情報を登録している点である。

【0069】

<データ4>

次に各種データについて説明する。

図15は、一括登録情報24の具体的な内容を示す図である。一括登録情報24の各メソッドは、ユーザによりコンパイルすべきメソッドとして指定されたメソッドや、プログラム実行シミュレーションの結果、頻繁に呼び出されることがわかっているメソッド等を想定している。

【0070】

図16は、キュー記憶部16Cに登録されているコンパイル要求情報の具体例を示す図である。同図に示すように、優先度高の第一キュー、優先度中の第二キュー、優先度低の第三キューが設けられており、第一キューには、図15で示した一括登録情報24に記載されていたメソッドm4、メソッドm5、メソッドm6それぞれのコンパイル要求情報が登録されている。

【0071】

また、第二キューには、優先度が付けられている各種メソッドが優先度順で登録されている。そして、第三キューには、第一キュー及び第二キューに登録されることのないメソッドのコンパイル要求情報が順番入替処理により動的に順番を変えながら配列されている。

<実施形態5>

実施形態5のプログラム実行制御装置は、関連メソッドテーブルをクラスファイルから取得し、コンパイル処理部6においてコンパイルを行うメソッドと関連付けされているメソッドがないかどうかを関連メソッドテーブルから検索し、関連付けされているメソッドが検出されれば、その検出されたメソッドの登録要求を行う。

【0072】

以下、実施形態5のプログラム実行制御装置について、実施形態1で説明したプログラム実行制御装置1とは異なる部分のみを、構成、データ、動作の各項に分けて説明する。

＜構成 5＞

図 17 は、実施形態 5 のプログラム実行制御装置 1D の機能構成図である。

【0073】

実施形態 1 のプログラム実行制御装置 1 と異なる点は、関連メソッド検索処理部 26 を備えている点である。

関連メソッド検索処理部 26 は、関連メソッドテーブル 28 を取得し、コンパイル要求取得部 17 から通知されたメソッドと関連付けされているメソッドがないかどうかを、取得した関連メソッドテーブル 28 から検索する検索部 27 とを有する。関連するメソッドが検出されれば、検出されたメソッドをコンパイル要求されたものとして、登録要求受付部 14D に通知する。

【0074】

＜データ 5＞

図 18 は、関連メソッドテーブル 28 の具体例を示す図である。

メソッド init と関連付けられているメソッドは、メソッド m1、メソッド m2 であり、メソッド m3 と関連付けられているメソッドは、メソッド m4、メソッド m5 である。

【0075】

＜動作 5＞

次に、関連メソッドの検索から登録までの処理である関連メソッド検索処理の動作について説明する。

図 19 は、関連メソッド検索処理部 26 においてなされる関連メソッド検索処理のフローを示す図である。

【0076】

まず、コンパイル要求取得部 17 からメソッドが通知されると、検索部 27 は、関連メソッドテーブル 28 を参照して、関連メソッドがあるかないかを検索する（ステップ S31）。

関連メソッドが検出されれば（ステップ S32：YES）、ステップ S33 に進む。検出されなければ（ステップ S32：NO）、関連メソッド検索処理を終了する。

【0077】

ステップS33において、検出された関連メソッドについて、コンパイル要求がなされたものとして、登録要求受付部14Dに通知し、関連メソッド検索処理を終了する。

ここで、具体的な動作を図18を用いて説明する。関連メソッド検索処理部26は、コンパイル要求取得部17からメソッドinitが通知されると、関連メソッドテーブル28を参照して、関連メソッドがあるかないかを検索する。その結果、図18に示すように、メソッドinitには、メソッドm1、メソッドm2が関連付けられているため、メソッドm1、メソッドm2を登録要求受付部14Dに通知する。

<実施形態6>

実施形態6のプログラム実行制御装置は、実施形態3で説明したように、各メソッドの優先度に基づいて登録位置決定処理を行うが、更に、コンパイルするメソッドの優先度と命令実行タスクの優先度との比較を行い、その結果、コンパイルするメソッドの優先度の方が命令実行タスクの優先度より高ければ、当該メソッドの優先度をコンパイルタスクの優先度とする優先度継承処理を行う。

【0078】

以下、実施形態6のプログラム実行制御装置について、実施形態3で説明したプログラム実行制御装置1Bとは異なる部分のみを、構成、動作の各項に分けて説明する。

<構成6>

図20は、実施形態6のプログラム実行制御装置1Eの機能構成図である。

【0079】

実施形態3のプログラム実行制御装置1Bと異なる点は、優先度継承部29を備えている点である。

優先度継承部29は、機能的には優先度比較部30とタスク優先度変更部31を含む。

優先度比較部30は、コンパイル要求取得部17からコンパイル開始するメソッドの優先度の通知を受けて、当該メソッドの優先度と命令実行タスクの優先度

とを比較する。

【0080】

比較した結果、コンパイル開始するメソッドの優先度の方が高い場合、タスク優先度変更部31は、マルチタスク制御部19Eに当該メソッドの優先度を、コンパイルタスクの優先度とする指示を行う。

マルチタスク制御部19Eは、タスク優先度変更部31からの指示に基づいて、コンパイルタスクの優先度を変更する。

【0081】

<動作6>

次に優先度継承処理について説明する。

図21は、優先度継承処理の動作を説明するためのフロー図である。

まず、優先度継承部29は、コンパイル要求取得部17からコンパイル開始するメソッドの優先度の通知を受けて、優先度比較部30において当該メソッドの優先度と命令実行タスクの優先度とを比較する（ステップS41）。

【0082】

その結果、メソッドの優先度の方が命令実行タスクの優先度より高ければ（ステップS42：YES）、タスク優先度変更部31は、コンパイルタスクの優先度をメソッドの優先度に変更する指示をマルチタスク制御部19Eに通知する（ステップS43）。命令実行タスクの優先度の方がメソッドの優先度より高ければ（ステップS42：NO）、ステップS41に戻る。

【0083】

例えば、命令実行タスクの優先度のデフォルト値が6で、コンパイルタスクの優先度のデフォルト値が2であり、コンパイルするメソッドの優先度が7である場合、優先度比較部30においてコンパイルするメソッドの優先度と命令実行タスクの優先度を比較すると、 $7 > 6$ より、コンパイルするメソッドの優先度の方が高いので、タスク優先度変更部31は、コンパイルタスクの優先度をメソッドの優先度に変更する指示をマルチタスク制御部19Eに対して行う。マルチタスク制御部19Eは、指示に基づいてコンパイルタスクの優先度を7とし、これは、命令実行タスクのデフォルト優先度6より高い値なので、コンパイルタスクに

対してプロセッサ資源を優先的に割り当てる。

<補足>

以上、本発明に係るプログラム実行制御装置について、実施形態1～6に基づいて説明したが、本発明はこれらの実施形態に限られないことは勿論である。即ち、

(1) 実施形態1の説明では、メソッド種別判定部12において、呼び出すメソッドがコンパイル済でなければ、直ちに当該メソッドのインタプリタ処理を行っていたが、呼び出すメソッドがコンパイル中であれば、そのコンパイル終了を待って、コンパイルされたメソッドのネイティブコードを実行するようにしてもよい。その場合の動作について図を用いて説明する。

【0084】

図22は、呼び出すメソッドがコンパイル中であれば待つ場合の命令実行タスクのフローを示す図である。

まず、命令実行部11は、取り出された命令をプロセッサに実行させる(ステップS51)。実行する命令がプログラム終了命令であれば(ステップS52: YES)、動作を終了する。実行する命令がプログラム終了命令でなければ(ステップS52: NO) ステップS53に進む。

【0085】

ステップS53において、実行する命令が呼出コードであれば(ステップS53: YES)、ステップS54に進む。ステップS53において、実行する命令が呼出コードでなければ(ステップS53: NO)、ステップS51に戻る。

ステップS54では、メソッド種別判定部12が、コンパイル済メソッド情報テーブルを参照して呼出対象のメソッドがコンパイル済みか否かを判定する。呼出対象のメソッドがコンパイル済であれば(ステップS54: YES)、ステップS55に進む。呼出対象のメソッドがコンパイル済でなければ(ステップS54: NO)、ステップS56に進む。

【0086】

ステップS55では、ネイティブコード取出部13が該当メソッドのネイティブコードをネイティブコード記憶部5から取り出し、取り出したネイティブコー

ドを命令実行部 11 に送る。その後、ステップ S 51 に戻る。

ステップ S 56 では、メソッド種別判定部 12 は、キュー制御部 15 にキューを参照させて、呼出対象のメソッドが現在コンパイル中か否かを確認させる。その結果、呼出対象のメソッドが現在コンパイル中であれば（ステップ S 56：YES）、コンパイルが終了するまで待ち（ステップ S 57）、そうでなければ（ステップ S 56：NO）、コンパイル要求をコンパイル要求管理部 4 に通知する。続いて、ステップ S 59 では、命令取出部 9 が、該当メソッドのバイトコードを取り出し、命令解読部 10 が、取り出したバイトコードの解読を行う。その後、ステップ S 51 に戻る。

（2）図 1 では、1 つの命令実行タスクと 1 つのコンパイルタスクの実行状態推移を示したが、命令実行タスク及びコンパイルタスクは複数であってもよく、この場合、各タスクにそれぞれ異なる優先度が付けられていてもよい。

【0087】

上記各実施形態では、コンパイルタスクの優先度より命令実行タスクの優先度を高くしていたが、複数のコンパイルタスクを設ける場合、特定のコンパイルタスクを例外として、当該コンパイルタスクの優先度を命令実行タスクの優先度より高く設定していてもよい。

例えば、実施形態 4 では、優先度が高、中、小と付けられている 3 つのキューを設けていたが、それぞれ優先度が異なるコンパイルタスクのキューであるとして、優先度高の第一キューに登録されているコンパイル要求情報のメソッドをコンパイルするコンパイルタスクの優先度を、命令実行タスクの優先度より高く設定しておくことが考えられる。

（3）上述した各実施形態のプログラム実行制御装置は、シングルプロセッサ資源の各タスクへの割り当てをスケジューリングして、各タスクを擬似並行処理しているものとして説明したが、本発明のプログラム実行制御装置は、マルチプロセッサ、或いは複数の独立したプロセッサを使用して、各タスクを同時に並行して実行するものであってもよい。

（4）本発明は、JIT コンパイラ付 JVM が実装されたプログラム実行制御装置の適用のみに限られず、コンパイラ機能を備えた、中間コードを解釈実行する

仮想マシンが実装されているプログラム実行制御装置全般に適用可能である。

(5) 各実施形態では、メソッドをコンパイルする順番をキューで管理していたが、本発明に係るプログラム実行制御装置は、必ずしもキューでコンパイルする順番を管理する必要はない。例えば、プログラム毎、或いはクラスファイル毎に予め設定されている閾値を取得して、コンパイル要求回数が閾値を超えたものからコンパイルするようにしてもよい。また、プログラム実行開始から所定時間毎に、その時に登録されている全てのコンパイル要求情報のうち、最も優先度が高いメソッドをコンパイルするようにしてもよいし、コンパイルタスクにプロセッサ資源が割り当てられる都度、その時に登録されているコンパイル要求情報のうち、最も優先度の高いメソッドをコンパイルするようにしてもよい。

(6) 実施形態4では、3つの優先度の異なるキューを用いていたが、3つに限られず、2つでも、4つでもよい。

(7) 本発明に係るプログラム実行制御装置は、一般的なJVMが備えているガーベジコレクション機能を有しており、ガーベジコレクションは、ガーベジコレクションタスクとして命令実行タスク及びコンパイルタスクと並行して実行されることを想定する。コンパイルしたネイティブコードのうち、再び取り出されることのないもの、使用頻度が小さいものについては、ガーベジコレクションによって破棄されるようにしてもよい。

(8) 各実施形態に示したプログラム実行制御装置による処理手順(図6、図9、図13、図19、図21、図22に示した手順等)を、プログラム実行機能を備える機器等を実行させるための制御プログラムを、記録媒体に記録し又は各種通信路等を介して、流通させ頒布することもできる。このような記録媒体には、ICカード、光ディスク、フレキシブルディスク、ROM等がある。流通、頒布された制御プログラムは、ROMを備える機器等にインストールされることにより利用に供され、その機器等はその制御プログラムの実行により各実施形態で示したようなプログラム実行制御装置の機能を実現する。

【0088】

【発明の効果】

以上の説明から明らかなように、本発明に係るプログラム実行制御装置は、バ

イトコード列を呼び出す呼出コードを含む1又は複数のバイトコード列からなるプログラムをプロセッサに実行させるプログラム実行制御装置であって、前記プログラムの実行過程において呼出コードが実行対象とされる度に、呼び出されるべきバイトコード列が前記プロセッサ用のネイティブコードに変換済みか否かを判定する判定手段と、前記判定手段により否定的な判定がなされた場合に、前記呼び出されるべきバイトコード列を逐次解釈して前記プロセッサに実行させ、且つ、当該バイトコード列を当該プロセッサ用のネイティブコードに変換する要求を行う第1手段と、前記判定手段により肯定的な判定がなされた場合に、前記ネイティブコードをプロセッサに実行させる第2手段と、前記第1手段によりなされるバイトコード列の逐次解釈実行又は前記第2手段によりなされる前記ネイティブコードの実行と並行して、前記要求に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させる第3手段とを備えることを特徴とする。

【0089】

この構成によると、呼び出すバイトコード列（メソッド）が未コンパイルであれば、当該バイトコード列を逐次解釈して実行するので、従来のJITコンパイラ付JVMが未コンパイルのメソッド実行時に当該メソッドをコンパイルすることにより生じていた、プログラム実行開始当初に顕著に現れる実行速度の低下を解消することができ、また、並行して未コンパイルメソッドのコンパイルも行うので、全体的なプログラム実行速度が低下することもない。

【0090】

また、前記プログラム実行制御装置は、マルチタスクオペレーティングシステムの制御下で、前記第1手段によりなされる前記バイトコード列の実行又は前記第2手段によりなされる前記ネイティブコードの実行と、前記第3手段によりなされる前記変換処理の実行とは、それぞれ異なるタスクとして実行され、前記第1手段又は前記第2手段のタスク実行は、前記第3手段のタスク実行より優先度が高いことを特徴としてもよい。

【0091】

この構成によると、コンパイル実行タスクより命令実行タスクが優先的に実行

されることになるので、プログラム実行開始当初に顕著に現れる実行速度の低下を解消することができる。

また、前記プログラム実行制御装置は、前記第1手段又は前記第2手段のタスク実行において待ち状態が発生した場合、前記第3手段のタスク実行に切り替える切替手段を備えることを特徴としてもよい。

【0092】

この構成によると、命令実行タスクにおいて待ち状態が発生した場合にコンパイルタスクの実行に切り替えるので、コンパイル処理を速やかに実行することができる。コンパイル処理が速やかに実行されれば、命令実行におけるインタプリタ処理の回数が減り、全体的なプログラム実行速度を速めることができる。

また、前記プログラム実行制御装置は、前記第1手段によりなされる前記要求に応じて、当該要求に係るバイトコード列をネイティブコードに変換するための情報であるコンパイル要求情報を記憶手段に登録し管理する要求管理手段を備え、前記第3手段は、前記第1手段によりなされるバイトコード列の逐次解釈実行又は前記第2手段によりなされる前記ネイティブコードの実行と並行して、前記要求管理手段によって前記記憶手段に登録されているコンパイル要求情報に係るバイトコード列をネイティブコードに変換する変換処理をプロセッサに実行させることを特徴としてもよい。

【0093】

この構成によると、要求管理手段によって記憶手段に登録及び管理されているコンパイル要求情報に基づいてバイトコード列のコンパイルを実行することができる。

また、前記プログラム実行制御装置の前記要求管理手段は、前記要求を先に受けたものからの順番となるキューで各コンパイル要求情報を登録して管理し、前記第3手段は、前記キューの先頭位置にあるコンパイル要求情報から順番に前記変換処理を前記プロセッサに実行させることを特徴としてもよい。

【0094】

この構成によると、コンパイル要求を受けた順番でバイトコード列のコンパイルを行うことができる。

また、前記プログラム実行制御装置の前記要求管理手段は、前記第1手段によりなされる前記要求に係るバイトコード列のコンパイル要求情報が前記記憶手段に既に登録されている場合に、2重登録を行わないことを特徴としてもよい。

【0095】

この構成によると、既に登録されているコンパイル要求情報を2重登録しないので、重複して同じメソッドをコンパイル処理することを防ぐことができる。

また、前記プログラム実行制御装置の前記要求管理手段は、繰り返し前記要求がなされるバイトコード列については、その要求回数を計数し、計数した要求回数を当該バイトコード列のコンパイル要求情報に含めて前記記憶手段に記録する回数記録手段と、記録した各コンパイル要求情報の要求回数を比較して、要求回数が多いもの順となるように各コンパイル要求情報のキュー位置を入替える入替手段とを備えることを特徴としてもよい。

【0096】

この構成によると、変換要求の多いバイトコード列を先にコンパイルすることができるので、命令実行におけるインタプリタ処理の回数が減り、全体的なプログラム実行速度を速めることができる。

また、前記プログラム実行制御装置は、各バイトコード列の優先度を示す優先度情報を取得する優先度情報取得手段を更に備え、前記要求管理手段は、前記第1手段によりなされる前記要求に応じて、取得された優先度情報を参照し、当該要求に係るバイトコード列の優先度を特定する特定手段と、前記記憶手段に登録されている各コンパイル要求情報に係るバイトコード列の優先度と、前記特定手段により特定された優先度とを比較する比較手段と、比較した結果に基づいて、優先度の高いもの順の配列となるように、前記要求に係るバイトコード列のコンパイル要求情報のキュー位置を決定する決定手段とを備えることを特徴としてもよい。

【0097】

この構成によると、優先度の高いバイトコード列から先にコンパイルすることができる。

また、前記プログラム実行制御装置は、前記要求管理手段は、優先度の異なる

複数のキューを管理しており、前記第3手段は、優先度が最も高いキューに登録されているコンパイル要求情報に係るバイトコード列から順番に前記変換処理を前記プロセッサに実行させることを特徴としてもよい。

【0098】

この構成によると、コンパイル要求情報を登録するキューで、優先的にコンパイルすべきバイトコード列とそうでないバイトコード列を区別することができる。

また、前記プログラム実行制御装置は、前記プログラムの実行前に、ネイティブコードに変換すべき複数のバイトコード列を示す特別要求情報を取得する特別要求情報取得手段を更に備え、前記要求管理手段は、取得された特別要求情報に示される各バイトコード列のコンパイル要求情報を、優先度が最も高いキューに一括登録し、管理することを特徴としてもよい。

【0099】

この構成によると、優先的にコンパイルすべきバイトコード列を特別要求情報に含めておくことで、これらのバイトコード列を優先的にコンパイルすることができる。

また、前記プログラム実行制御装置は、各バイトコード列の優先度を示す優先度情報を取得する優先度情報取得手段と、前記変換処理を行うバイトコード列の優先度と前記命令実行タスクの優先度とを比較する比較手段と、比較した結果、前記バイトコード列の優先度の方が命令実行タスクの優先度より高い場合に、前記コンパイルタスクの優先度を一時的に高くする優先度変更手段とを更に備えることを特徴としてもよい。

【0100】

この構成によると、命令実行タスクの優先度より高い優先度のバイトコード列をコンパイルする場合、一時的にコンパイルタスクの優先度を高くするので、当該バイトコード列についてはコンパイルを速やかに行うことができる。

また、前記プログラム実行制御装置は、あるバイトコード列と関連する関連バイトコード列を対応付けている対応情報を取得する対応情報取得手段と、前記対応情報を参照して、前記要求に係るバイトコード列と対応付けられている関連バ

イトコード列があるか否かを検索する検索手段とを更に備え、前記検索手段による検索結果、関連バイトコード列が検出された場合、前記要求管理手段は、検出された関連バイトコード列のコンパイル要求情報を記憶手段に登録することを特徴としてもよい。

【0101】

この構成によると、コンパイル要求されたバイトコード列と関連するバイトコード列のコンパイル要求情報を登録することができる。

また、前記プログラム実行制御装置は、前記判定手段により否定的な判定がなされたときに、前記バイトコード列が現在、前記変換処理の途中であるか否かを判定する第2判定手段と、前記第2判定手段により肯定的な判定がなされた場合に、当該変換処理が終了するのを待って、変換されたネイティブコードを前記プロセッサに実行させる第4手段とを更に備えることを特徴としてもよい。

【0102】

この構成によると、未コンパイルのバイトコード列の実行時に、当該バイトコード列がコンパイル中であれば、そのコンパイルが終了するのを待って、変換されたネイティブコードを実行することができる。

また、前記プログラム実行制御装置は、各バイトコード列の優先度を示す優先度情報を取得する優先度情報取得手段を更に備え、前記第3手段は、前記記憶手段に複数登録されているコンパイル要求情報に係るバイトコード列のうち、取得された前記優先度情報に基づいて優先度の高いものから順番に前記変換処理を前記プロセッサに実行させることを特徴としてもよい。

【0103】

この構成によると、優先度の高いバイトコード列から先にコンパイルすることができる。

また、前記プログラム実行制御装置の前記要求管理手段は、閾値を取得する取得手段と、繰り返し前記要求がなされるバイトコード列については、その要求回数を計数し、当該バイトコード列のコンパイル要求情報に含めて計数した要求回数を前記記憶手段に記録する回数記録手段とを備え、前記第3手段は、前記要求回数が閾値を超えたものから順番に前記変換処理を前記プロセッサに実行させる

ことを特徴としてもよい。

【0104】

この構成によると、閾値を超えるコンパイル要求回数となったバイトコード列からコンパイルすることができる。

【図面の簡単な説明】

【図1】

命令実行タスクとコンパイルタスクの実行状態推移を示す図である。

【図2】

実施形態1のプログラム実行制御装置1の機能構成図である。

【図3】

Java言語で記述されたソースプログラムの一例を示す図である。

【図4】

実施形態1のキュー記憶部16に登録されているコンパイル要求情報の具体例を示す図である。

【図5】

コンパイル済メソッド情報テーブルの一例を示す図である。

【図6】

命令実行処理を説明するためのフロー図である。

【図7】

実施形態2のプログラム実行制御装置1Aの機能構成図である。

【図8】

実施形態2のキュー記憶部16Aに登録されているコンパイル要求情報の具体例を示す図である。

【図9】

コンパイル要求情報の順番入替処理を説明するためのフロー図である。

【図10】

実施形態3のプログラム実行制御装置1Bの機能構成図である。

【図11】

クラスファイルから取得した優先度情報テーブル23の具体的な内容を示す図

である。

【図 12】

キュー記憶部 16 B に登録されているコンパイル要求情報の具体例を示す図である。

【図 13】

コンパイル要求情報の登録位置決定処理を説明するためのフロー図である。

【図 14】

実施形態 4 のプログラム実行制御装置 1 C の機能構成図である。

【図 15】

一括登録情報 24 の具体的な内容を示す図である。

【図 16】

キュー記憶部 16 C に登録されているコンパイル要求情報の具体例を示す図である。

【図 17】

実施形態 5 のプログラム実行制御装置 1 D の機能構成図である。

【図 18】

関連メソッドテーブル 28 の具体例を示す図である。

【図 19】

関連メソッド検索処理を説明するためのフロー図である。

【図 20】

実施形態 6 のプログラム実行制御装置 1 E の機能構成図である。

【図 21】

優先度継承処理を説明するためのフロー図である。

【図 22】

命令実行処理の変形例を説明するためのフロー図である。

【符号の説明】

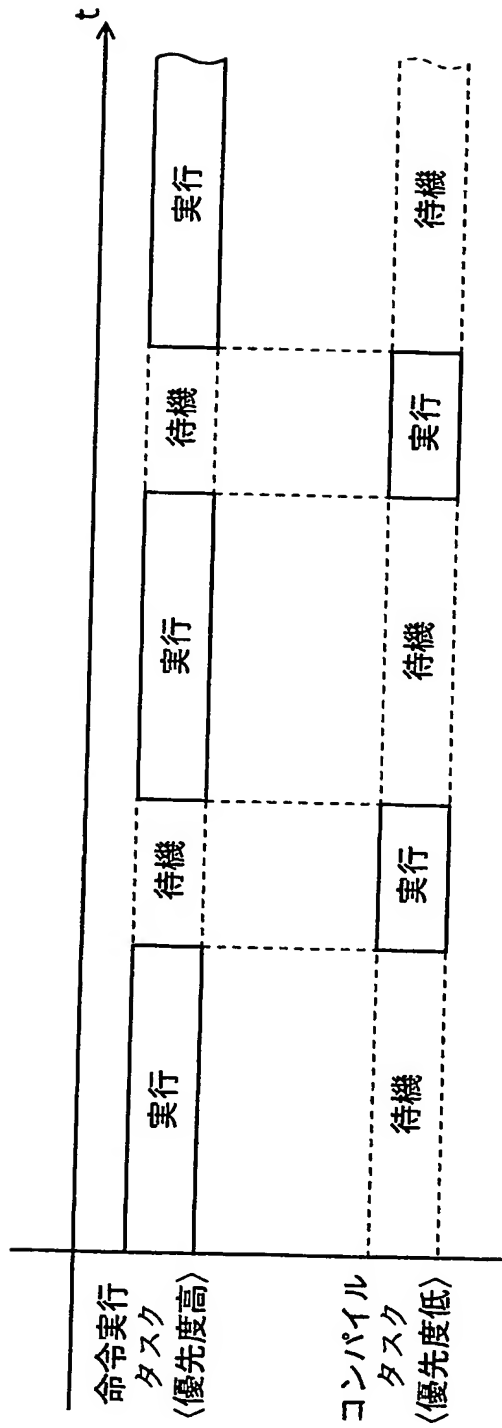
- 1、1 A、1 B、1 C、1 D、1 E プログラム実行制御装置
- 2 クラスロード記憶部
- 3、3 B、3 C、3 D、3 E 命令実行処理部

- 4、4 A、4 B、4 C、4 D、4 E コンパイル要求管理部
- 5 ネイティブコード記憶部
- 6 コンパイル処理部
- 7 コンパイル済メソッド情報テーブル
- 8、8 B、8 C、8 D、8 E クラスロード部
- 9 命令取出部
- 10 命令解読部
- 11 命令実行部
- 12 メソッド種別判定部
- 13 ネイティブコード取出部
- 14 登録要求受付部
- 15、15 A、15 B、15 C、15 D、15 E キュー制御部
- 16、16 A、16 B、16 C、16 D キュー記憶部
- 17 コンパイル要求取得部
- 18 メソッドコンパイル部
- 19、19 E マルチタスク制御部
- 20 ネットワーク
- 21 アンテナ
- 22 加算部
- 23 優先度情報テーブル
- 24 一括登録情報
- 25 一括登録部
- 26 関連メソッド検索処理部
- 27 検索部
- 28 関連メソッドテーブル
- 29 優先度継承部
- 30 優先度比較部
- 31 タスク優先度変更部

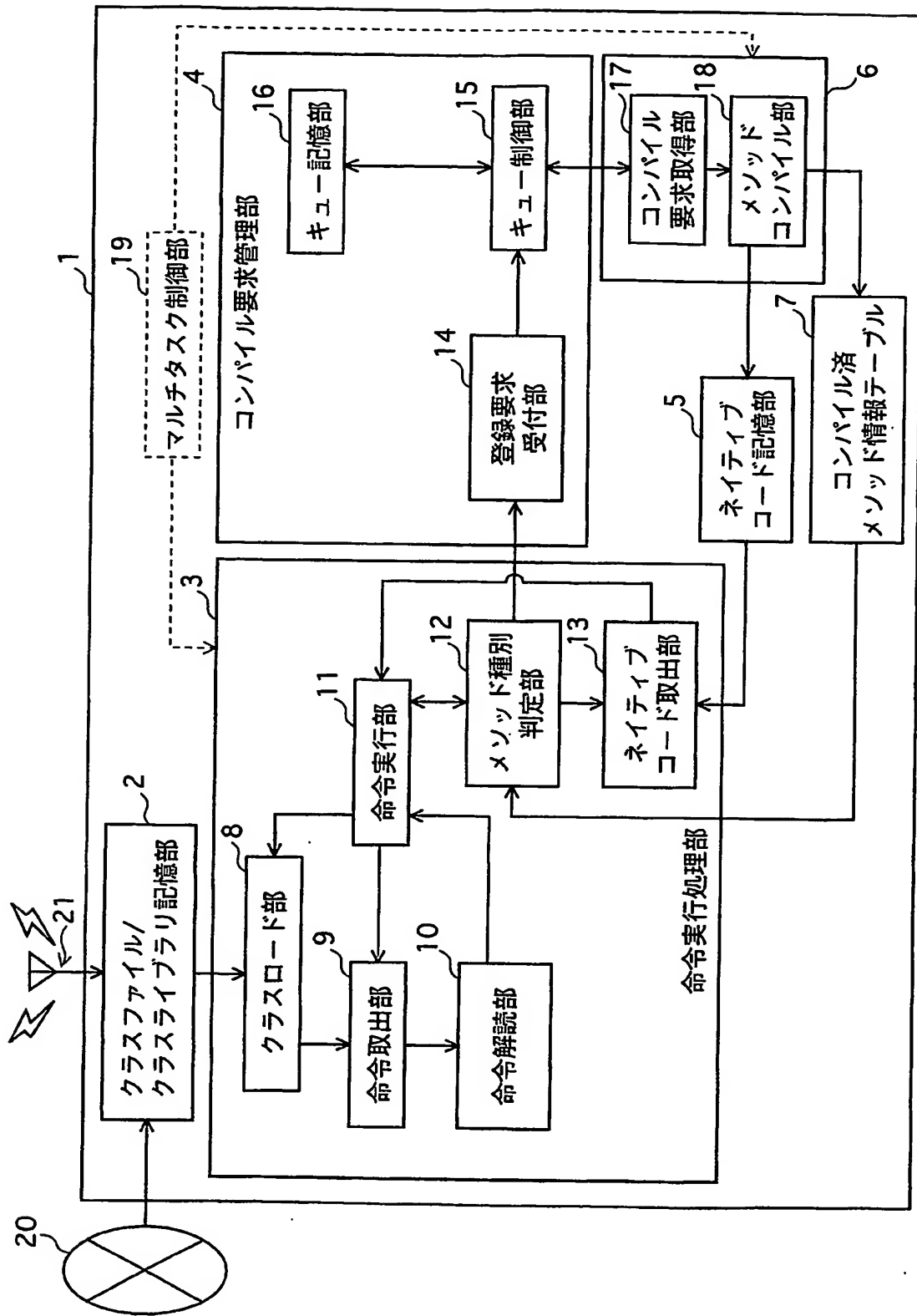
【書類名】

図面

【図 1】



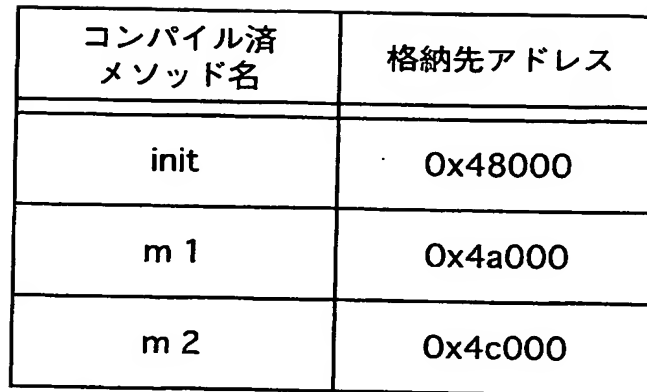
【図 2】



【図 3】

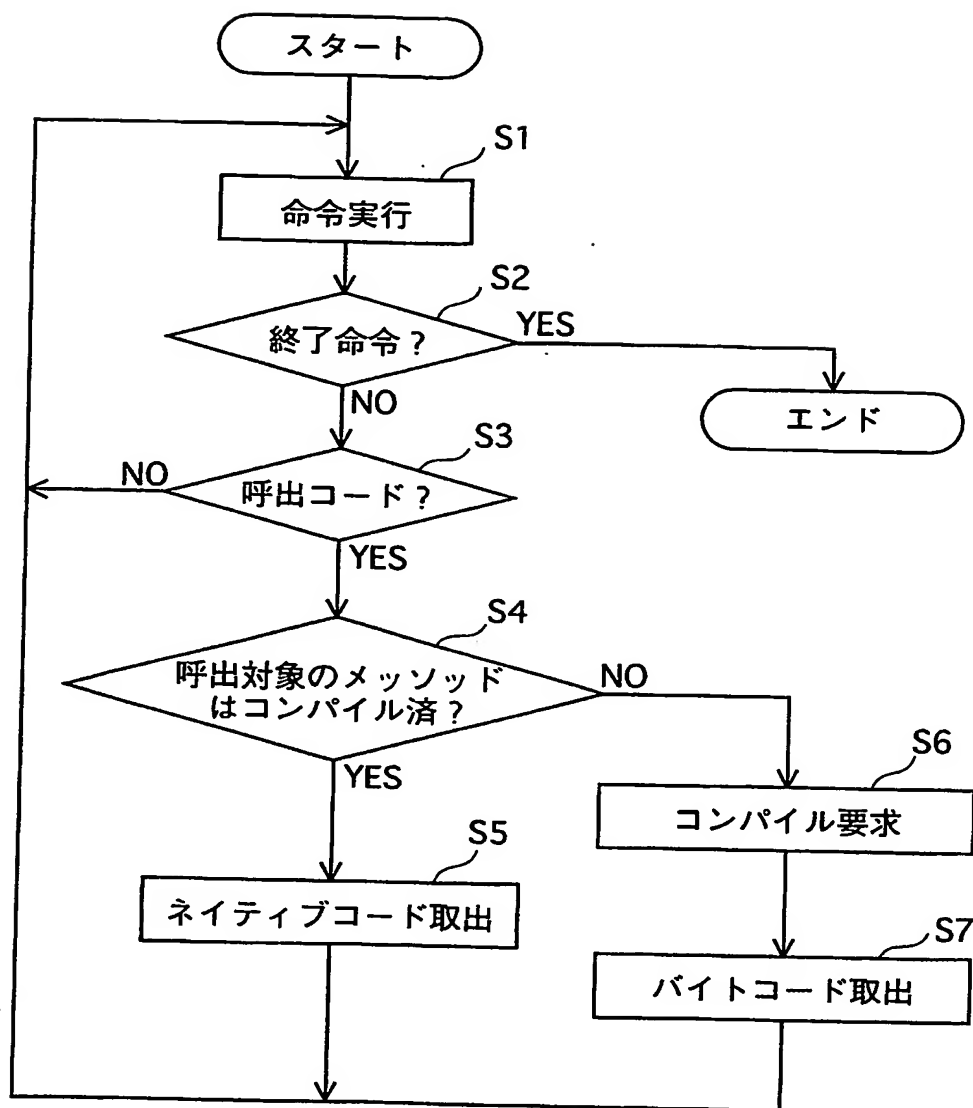
```
public class A {  
    public void init () {  
        int i;  
        for (i=0; i<10 ;i++) {  
            m1();  
        }  
  
        for (i=0; i<5 ;i++) {  
            m2();  
        }  
    }  
    public void m1() {  
        .....  
    }  
  
    public void m2() {  
        .....  
    }  
}
```


【図 5】

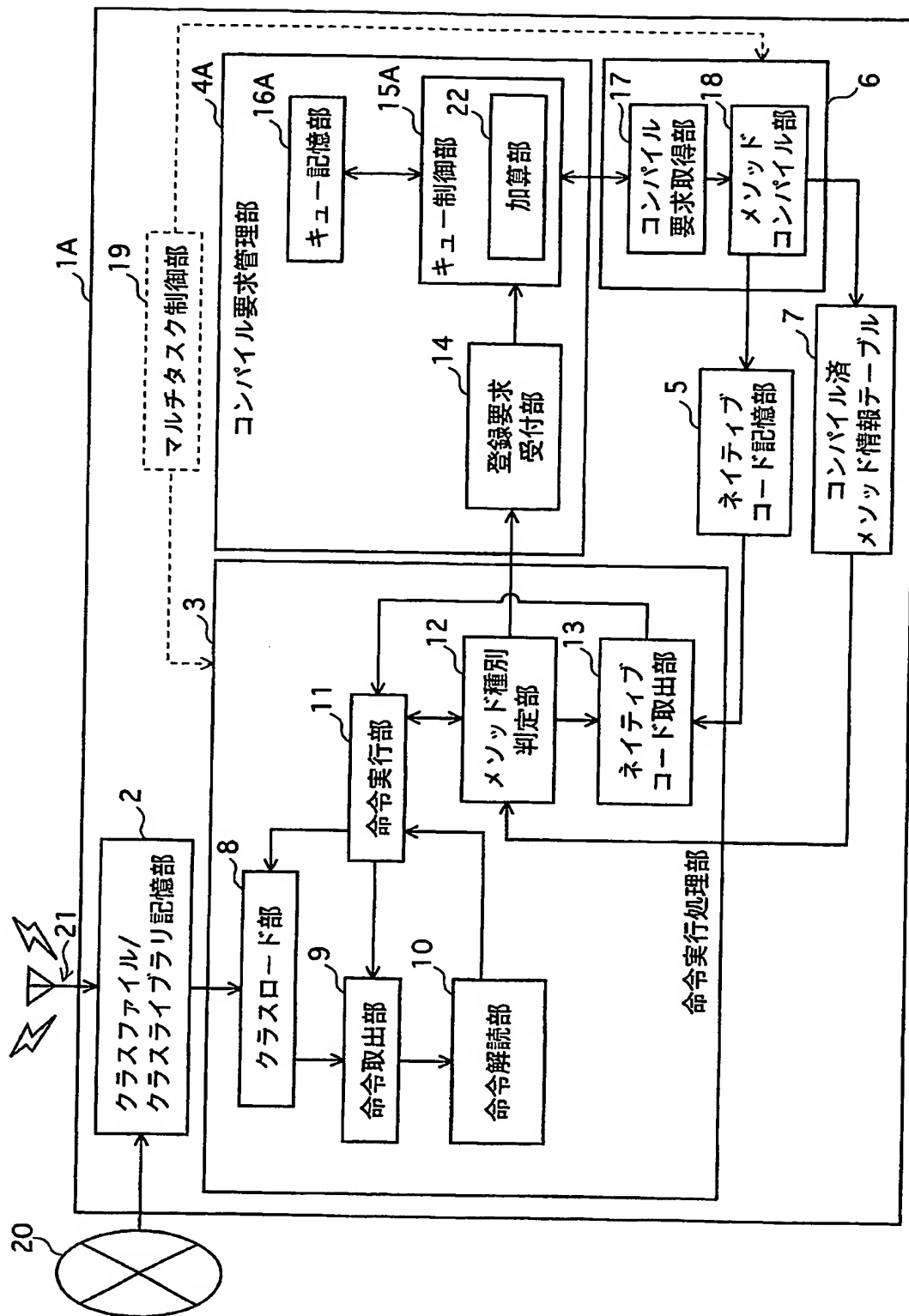


コンパイル済 メソッド名	格納先アドレス
init	0x48000
m 1	0x4a000
m 2	0x4c000

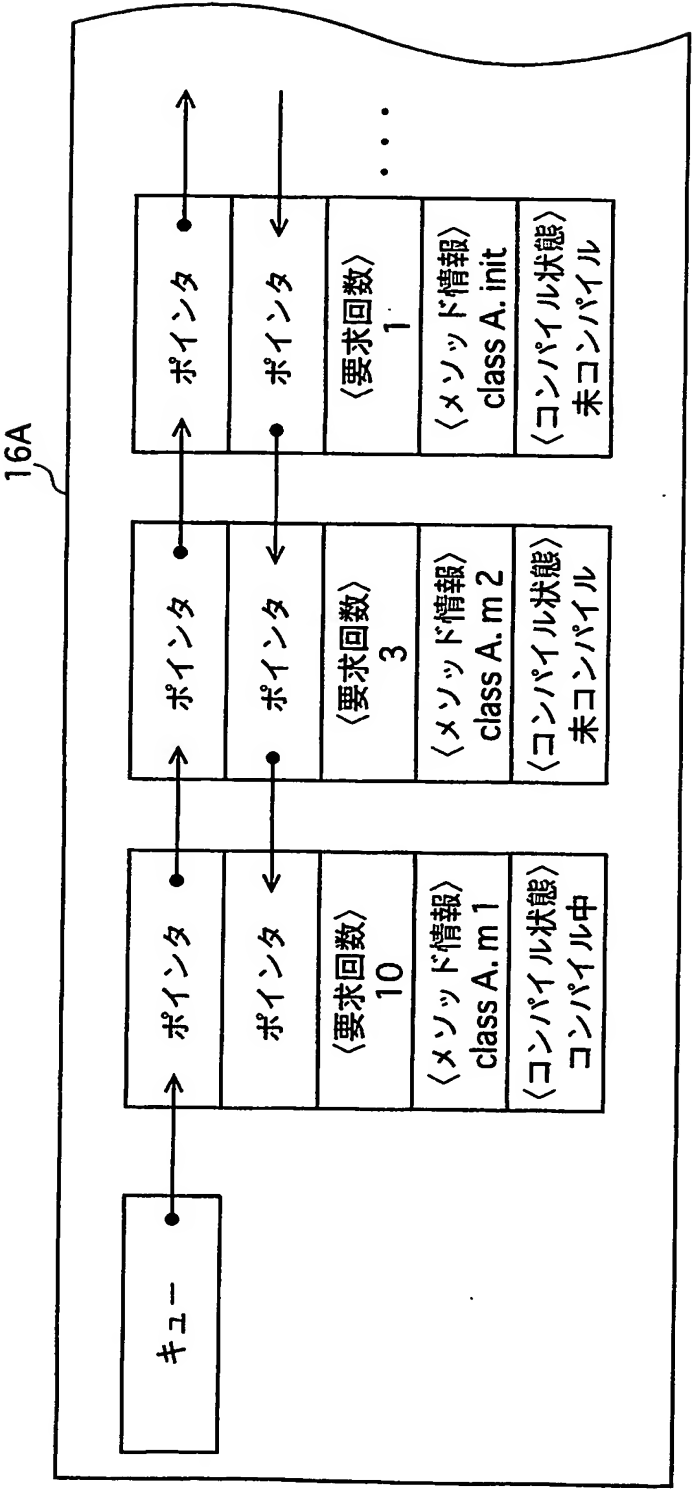
【図 6】



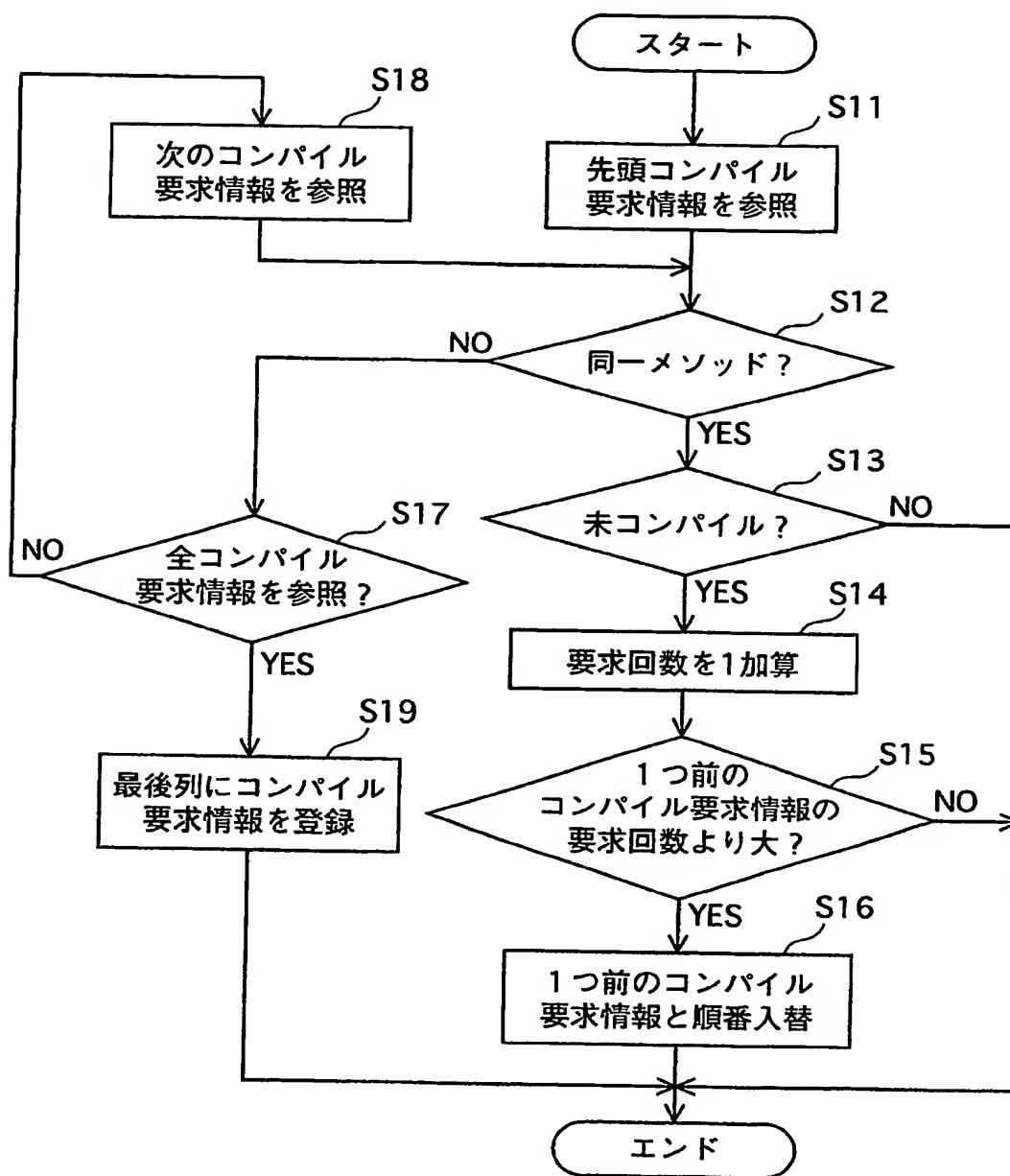
【図 7】



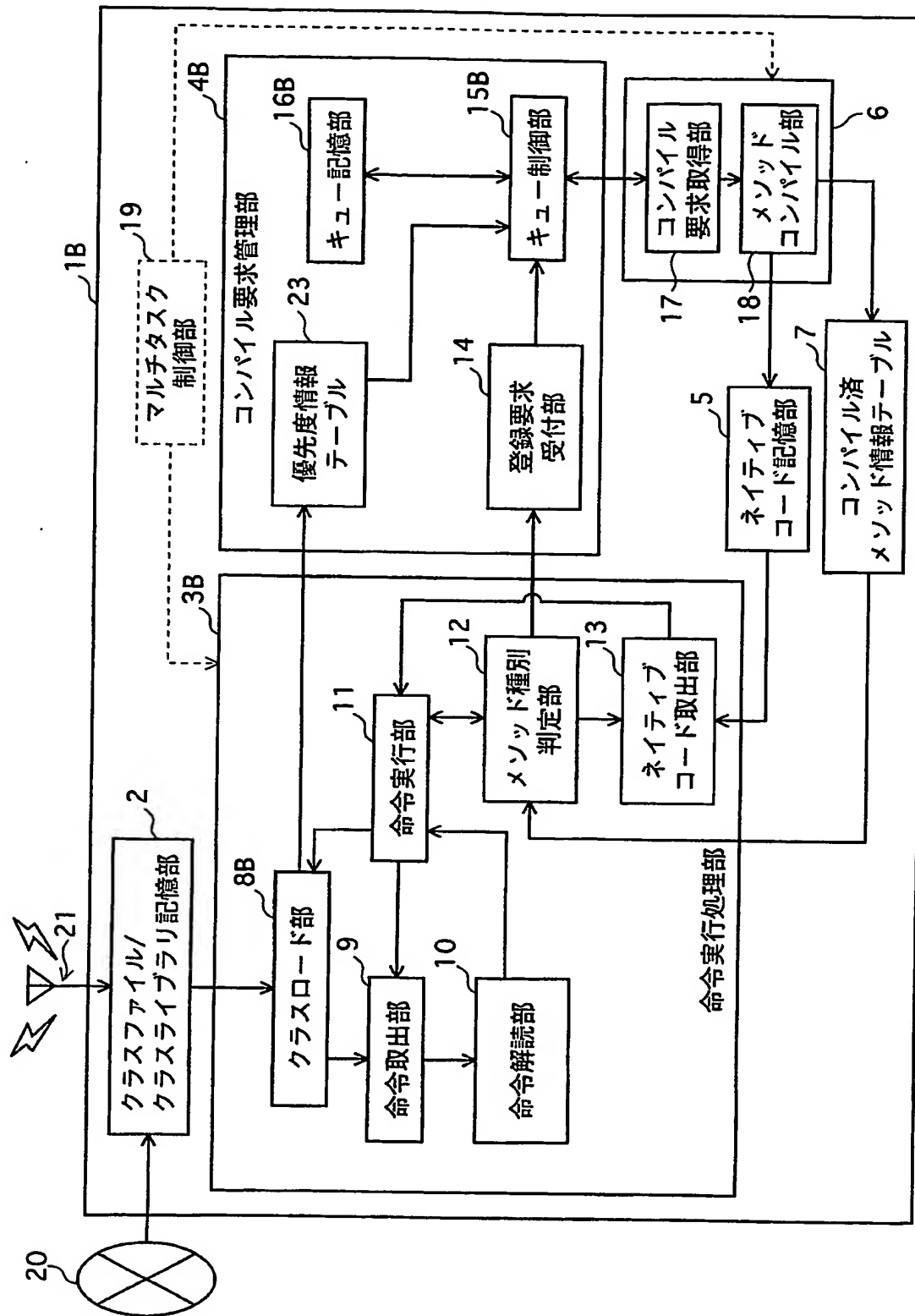
【図 8】



【図 9】



【図 10】

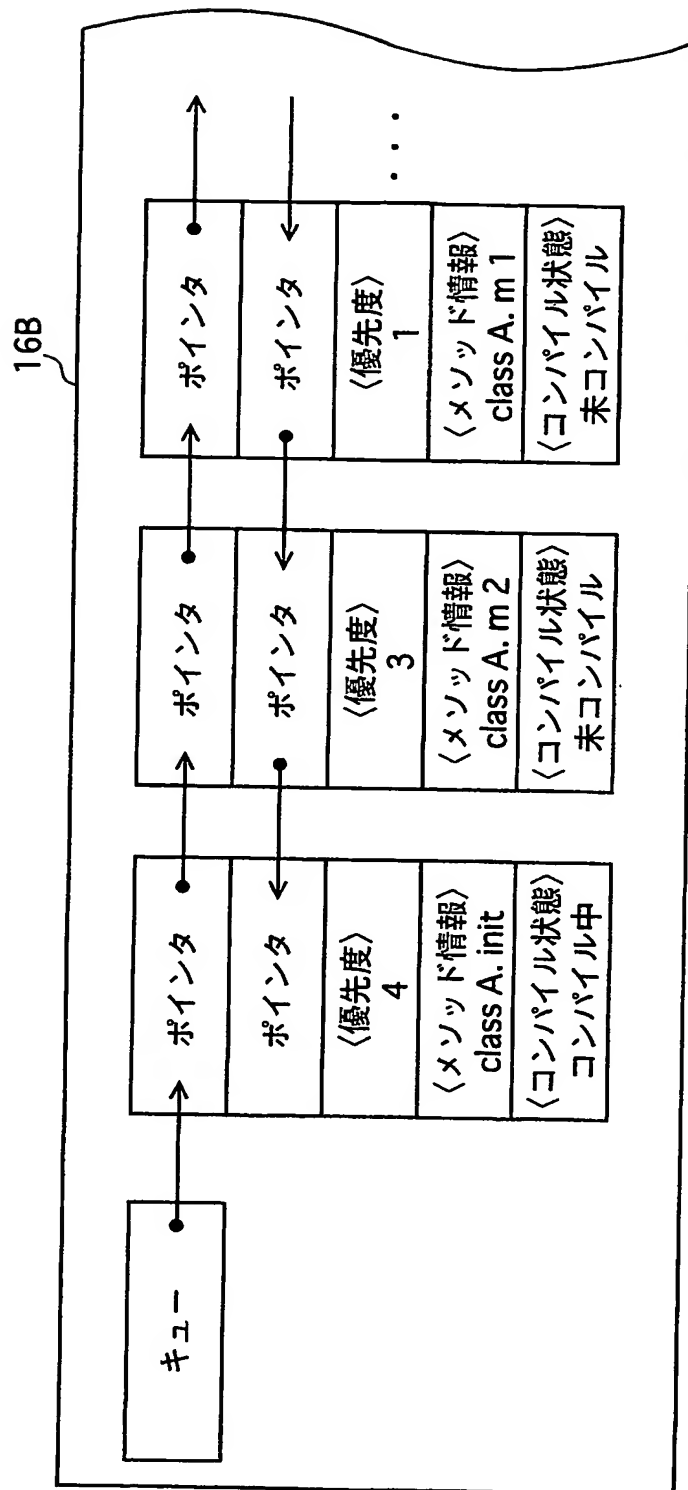


【図 11】

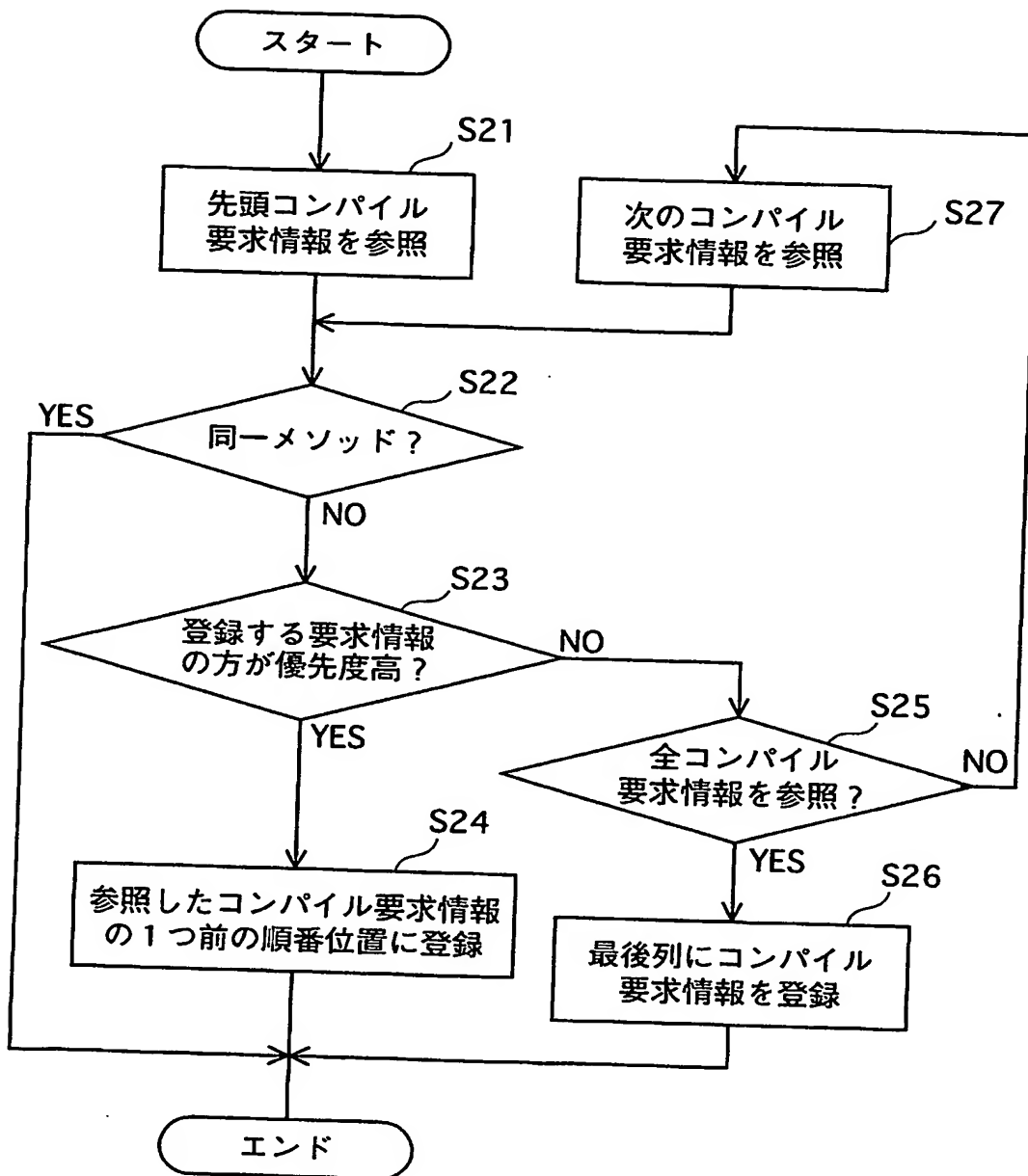
19

メソッド名	優先度
init	4
m 1	1
m 2	3
m 3	5

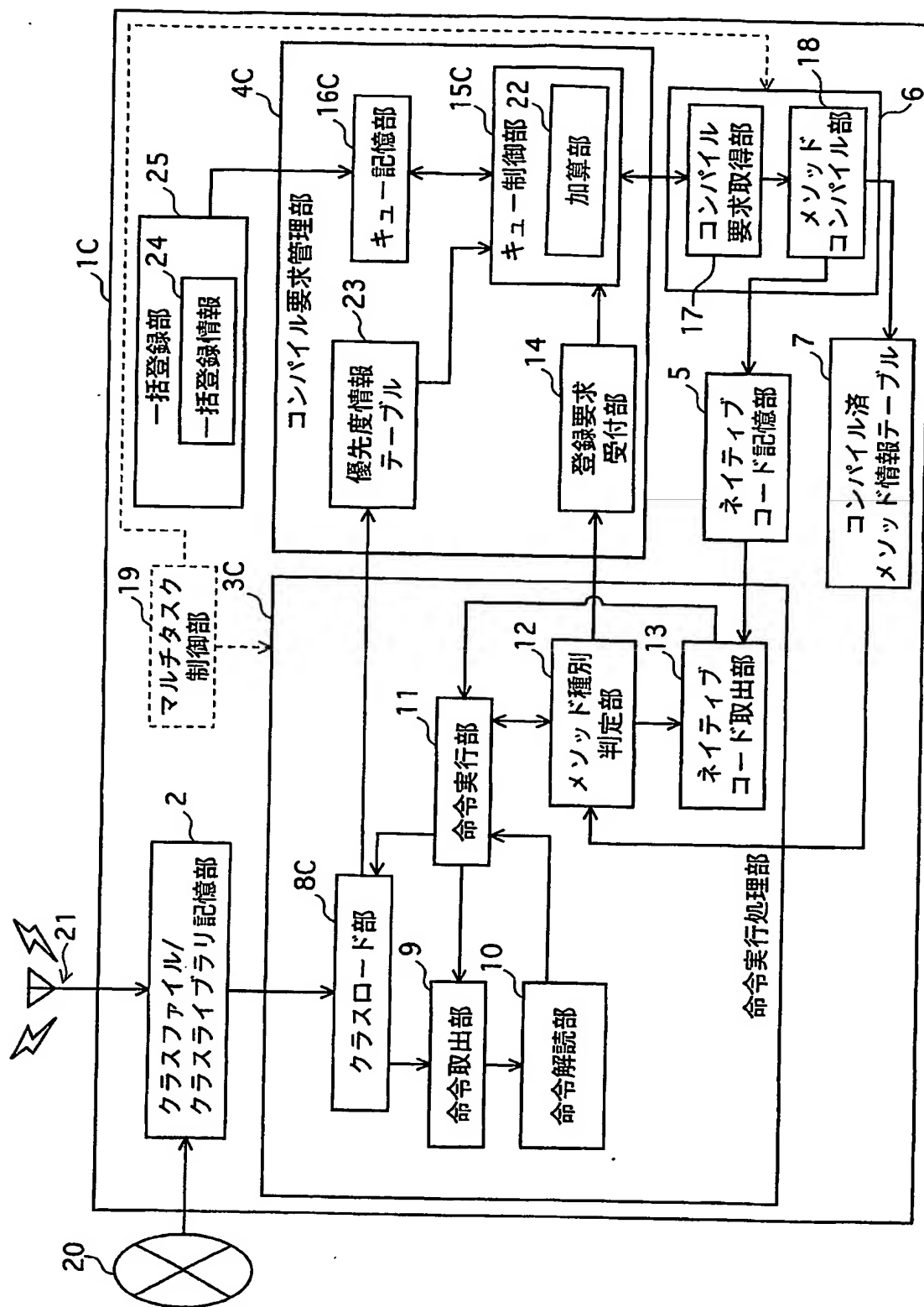
【図 12】



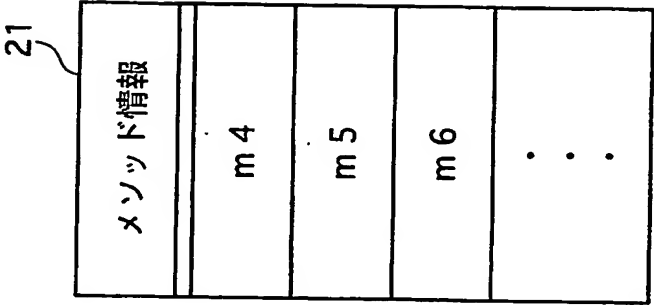
【図 13】



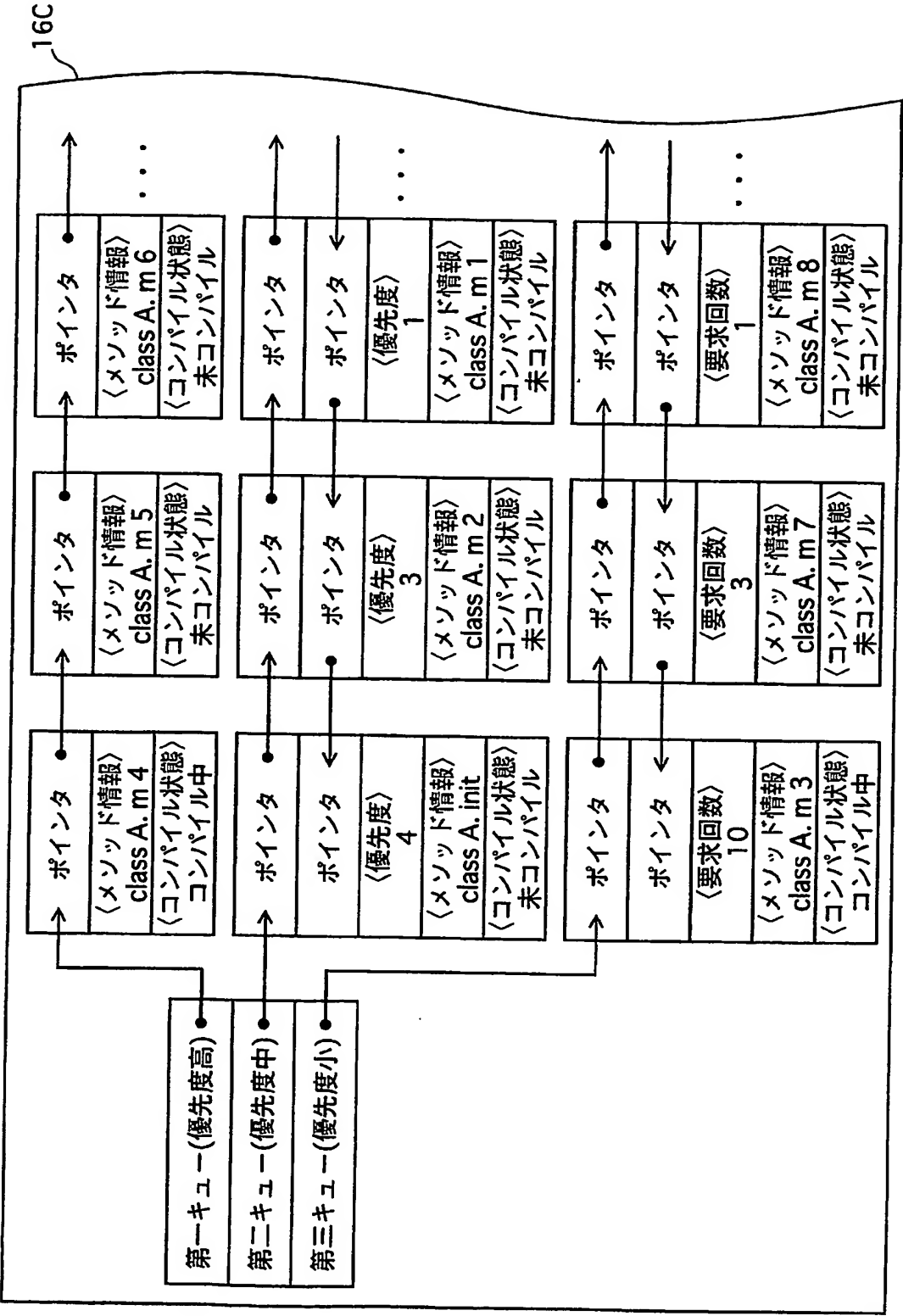
【図 14】



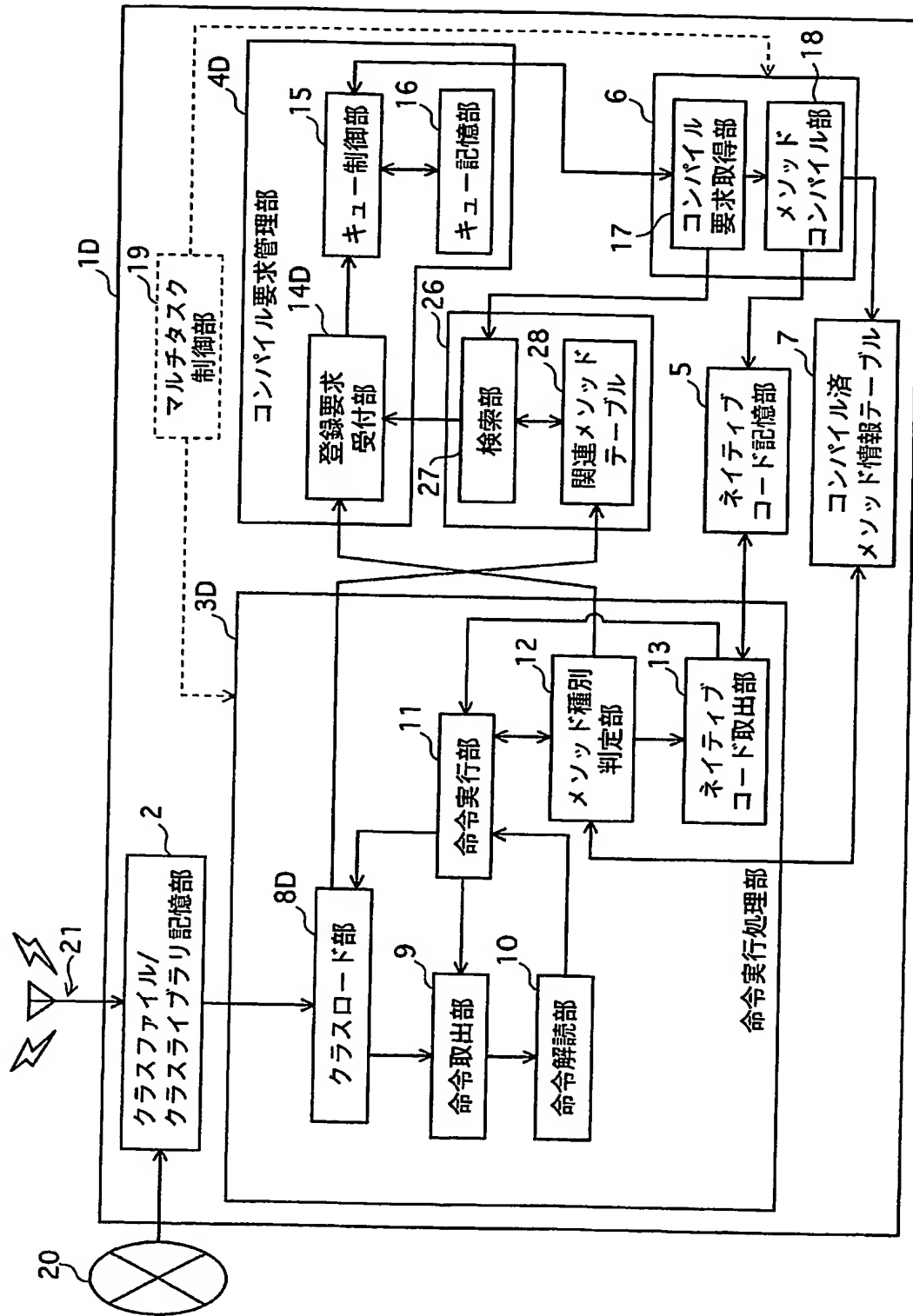
【図 15】



【図 16】



【図 17】

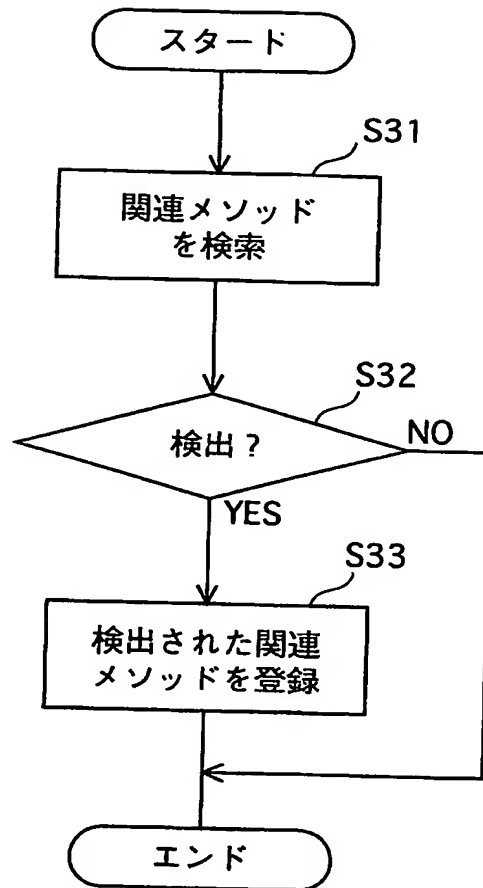


【図18】

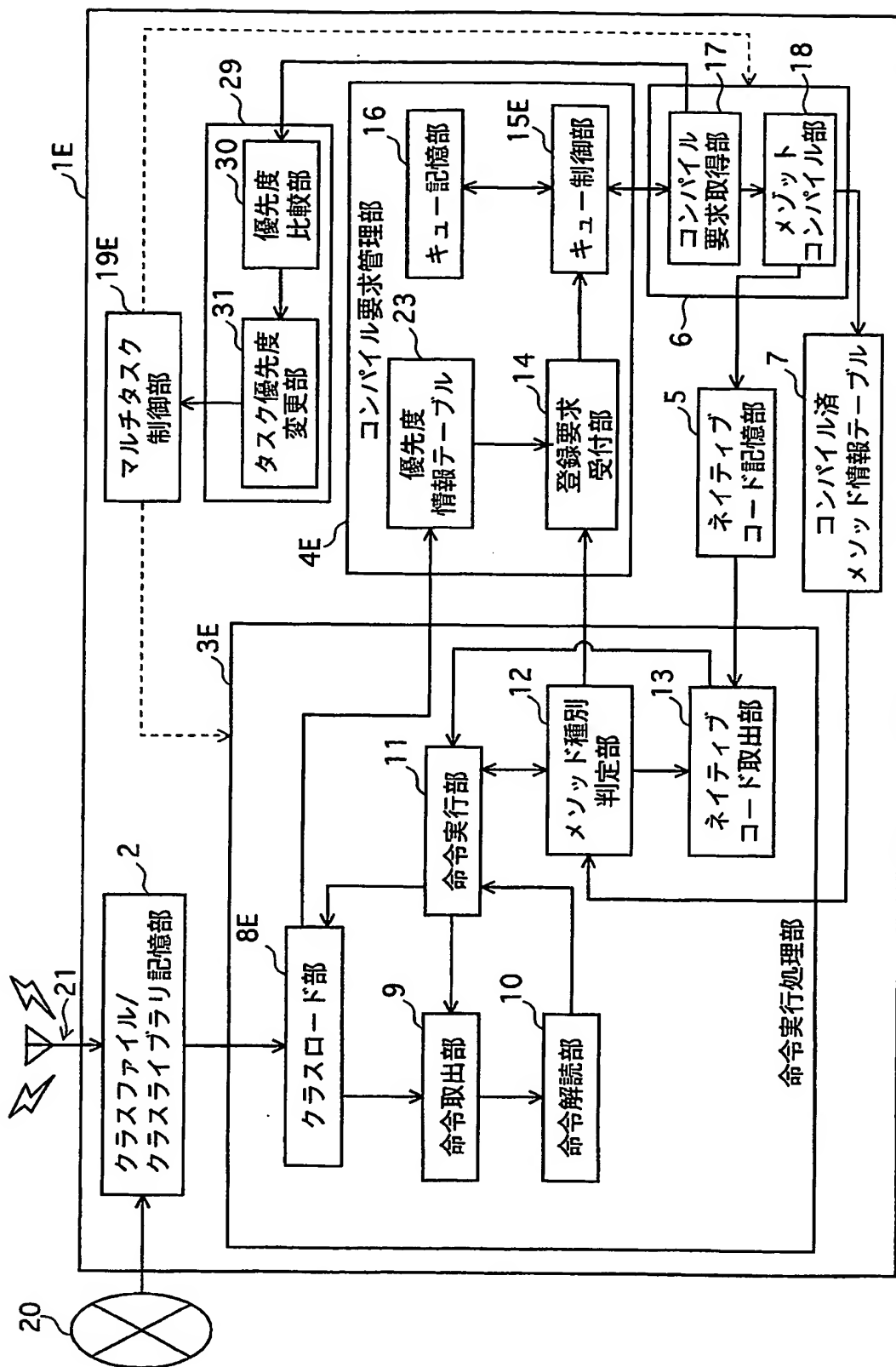
22

メソッド名	関連メソッド名1	関連メソッド名2	
init	m 1	m 2	...
m 3	m 4	m 5	...
⋮ ⋮ ⋮	⋮ ⋮ ⋮	⋮ ⋮ ⋮	⋮ ⋮ ⋮

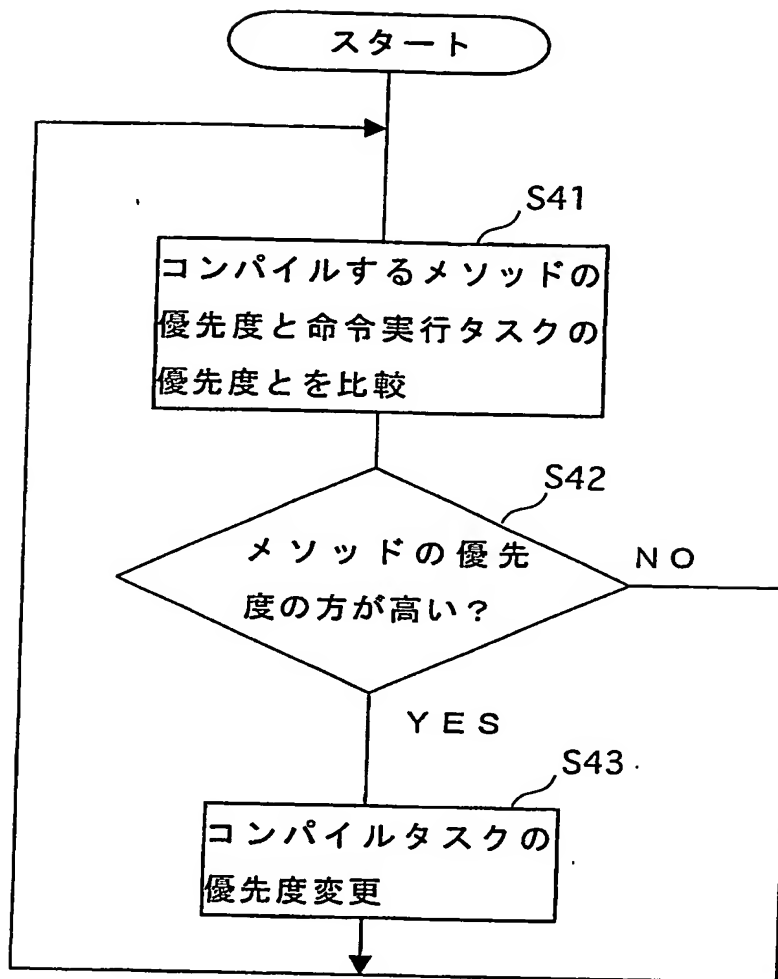
【図 19】



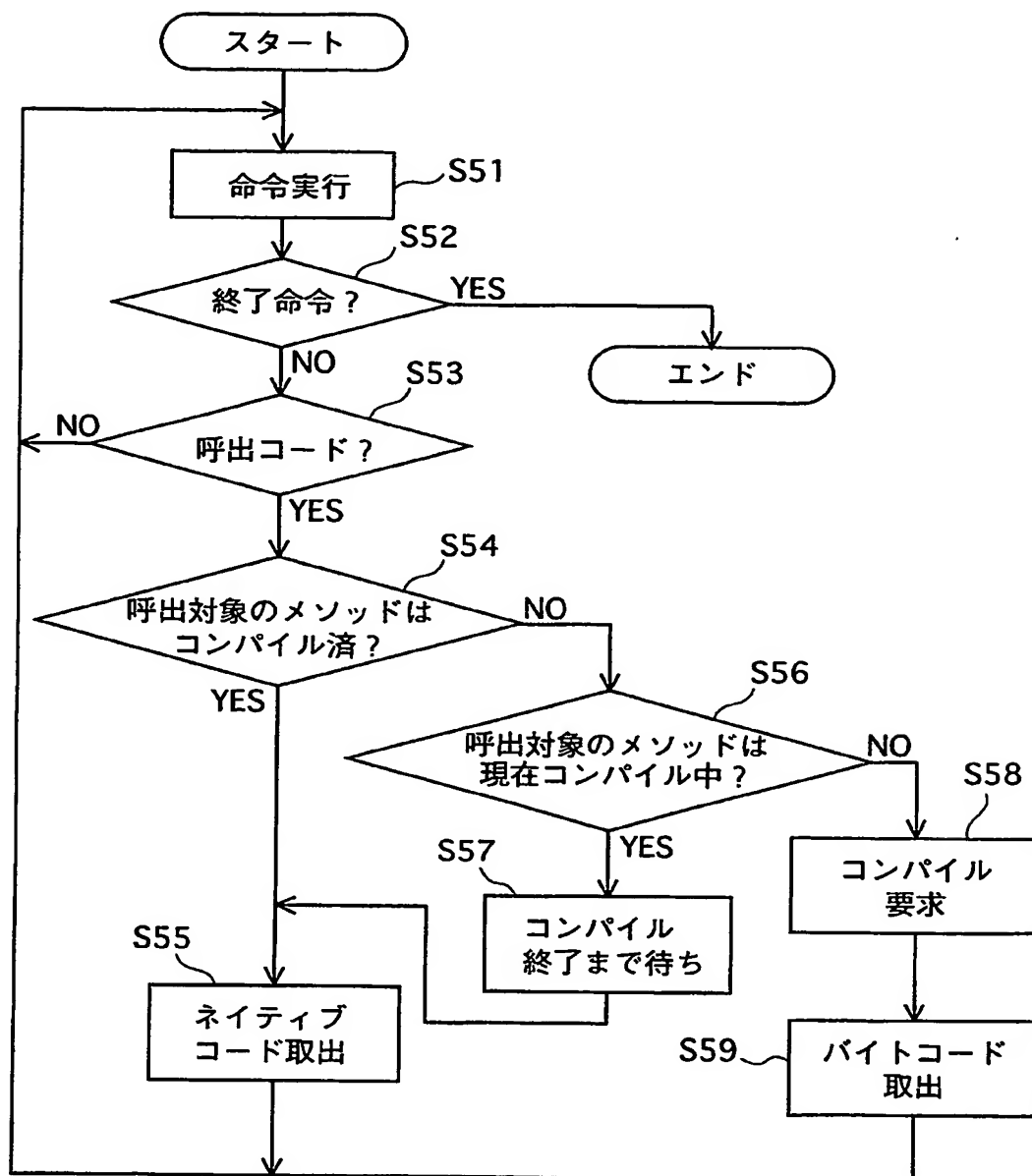
【図 20】



【図 21】



【図 22】



【書類名】 要約書

【要約】

【課題】 従来の J i T コンパイラ付 J V M が実装されたプロセッサを備える家電製品において、未コンパイルのメソッド実行時に当該メソッドをコンパイルすることにより生じていた実行速度の低下を解消するプログラム実行制御装置を提供する。

【解決手段】 本発明に係るプログラム実行制御装置は、プログラムの実行過程においてメソッドを呼び出す際に、当該メソッドが未コンパイルであるか否かを判定し、未コンパイルであれば、当該メソッドのバイトコードをインタプリタ処理で実行すると共に当該メソッドをコンパイルする要求を行い、コンパイル済であれば、当該メソッドのネイティブコードを実行する。コンパイル要求されたメソッドのコンパイルは、バイトコードのインタプリタ処理による実行又はネイティブコードの実行といった命令実行を行うタスクとは別のタスクで行う。

【選択図】 図 1

特願 2003-151472

ページ: 1/E

出願人履歴情報

識別番号

[000005821]

1. 変更年月日
[変更理由]

住所
氏名

1990年 8月28日

新規登録

大阪府門真市大字門真1006番地
松下電器産業株式会社